NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AUTOMATIC TARGET RECOGNITION: STATISTICAL FEATURE SELECTION OF NON-GAUSSIAN DISTRIBUTED TARGET CLASSES

by

Matthew J. Wilder

June 2011

Thesis Advisor:	Grace A. Clark
Second Reader:	Monique P. Fargues

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Automatic Target Recognition: Statistical Feature Selection of Non-Gaussian Distributed Target Classes		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Matthew J. Wilder			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N.A._____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE** A	

13. ABSTRACT (maximum 200 words)

Target and pattern recognition systems are in widespread use. Efforts have been made in all areas of pattern recognition to increase the performance of these systems. Feature extraction, feature selection, and classification are the major aspects of a target recognition system. This research proposes algorithms for selecting useful statistical features in pattern/target classification problems in which the features are non-Gaussian distributed. In engineering practice, it is common to either not perform any feature selection procedure or to use a feature selection algorithm that assumes the features are Gaussian distributed. These results can be far from optimal if the features are non-Gaussian distributed, as they often are. This research has the goal of mitigating that problem by creating algorithms that are useful in practice.

This work focuses on the performance of three common feature selection algorithms: the Branch and Bound, the Sequential Forward Selection, and Exhaustive Search algorithms. Ordinarily, the performance index used to measure the class separation in feature space involves assuming the data are Gaussian and deriving tractable performance indices that can be calculated without estimating the probability density functions of the class data. The advantage of this approach is that it produces feature selection algorithms that have low computational complexity and do not require knowledge of the data densities. The disadvantage is that these algorithms may not perform reasonably when the data are non-Gaussian. This research examines the use of information-theoretic class separability measures that can deal with the non-Gaussian case. In particular, this work shows that the Hellinger Distance (a type of divergence) has very desirable mathematical properties and can be useful for feature selection when accompanied by a suitable density estimator.

The suitable density estimator for this research is the multivariate kernel density estimator. In selecting the best feature subset of non-Gaussian distributed features, results show that the Hellinger distance outperformed the other class separability measures in several instances highlighted in this report.

14. SUBJECT TERMS			**15. NUMBER OF PAGES** 149
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AUTOMATIC TARGET RECOGNITION: STATISTICAL FEATURE SELECTION OF NON-GAUSSIAN DISTRIBUTED TARGET CLASSES

Matthew J. Wilder
Ensign, United States Navy
B.S.E.E, United States Naval Academy, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2011

Author: Matthew J. Wilder

Approved by: Grace A. Clark
Thesis Advisor

Monique P. Fargues
Second Reader

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Target and pattern recognition systems are in widespread use. Efforts have been made in all areas of pattern recognition to increase the performance of these systems. Feature extraction, feature selection, and classification are the major aspects of a target recognition system. This research proposes algorithms for selecting useful statistical features in pattern/target classification problems in which the features are non-Gaussian distributed. In engineering practice, it is common to either not perform any feature selection procedure or to use a feature selection algorithm that assumes the features are Gaussian distributed. These results can be far from optimal if the features are non-Gaussian distributed, as they often are. This research has the goal of mitigating that problem by creating algorithms that are useful in practice.

This work focuses on the performance of three common feature selection algorithms: the Branch and Bound, the Sequential Forward Selection, and Exhaustive Search algorithms. Ordinarily, the performance index used to measure the class separation in feature space involves assuming the data are Gaussian and deriving tractable performance indices that can be calculated without estimating the probability density functions of the class data. The advantage of this approach is that it produces feature selection algorithms that have low computational complexity and do not require knowledge of the data densities. The disadvantage is that these algorithms may not perform reasonably when the data are non-Gaussian. This research examines the use of information-theoretic class separability measures that can deal with the non-Gaussian case. In particular, this work shows that the Hellinger Distance (a type of divergence) has very desirable mathematical properties and can be useful for feature selection when accompanied by a suitable density estimator.

The suitable density estimator for this research is the multivariate kernel density estimator. In selecting the best feature subset of non-Gaussian distributed features, results show that the Hellinger distance outperformed the other class separability measures in several instances highlighted in this report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Target and pattern recognition systems are growing in widespread use. Efforts have been made in all areas of pattern recognition to increase the performance of these systems. Feature extraction, feature selection, and classification are the major aspects of a target recognition system. This research proposes creating algorithms for selecting useful statistical features in pattern/target classification problems in which the features are non-Gaussian distributed. In engineering practice, it is common to either not perform any feature selection procedure or to use a feature selection algorithm that assumes the features are Gaussian distributed. These results can be far from optimal if the features are non-Gaussian distributed, as they often are. This research has the goal of mitigating that problem by creating a useful feature selection algorithm that can be used in practice.

The approach is to focus on the performance of three common feature selection algorithms: the Branch and Bound, the Sequential Forward Selection, and the Exhaustive Search algorithms. Ordinarily, the performance index used to measure the class separation in feature space involves assuming the data are Gaussian and deriving tractable performance indices that can be calculated without estimating the probability density functions of the class data. The advantage of this approach is that it produces feature selection algorithms that have low computational complexity and do not require knowledge of the data densities. The disadvantage is that these algorithms may not perform reasonably when the data are non-Gaussian, as they commonly are in practice. This research examines the use of information-theoretic class separability measures that can deal with the non-Gaussian case. In particular, this work shows that the Hellinger Distance (a type of divergence) has very desirable mathematical properties and can be useful for feature selection when accompanied by a suitable density estimator.

The class data sets used included several simulated data sets and the classic Fisher iris data set for classification. The simulated data sets include the two Gaussian target classes case, the one Gaussian and one non-Gaussian case, and the two non-Gaussian classes case. The data sets were divided into training and test sets and were processed

with the feature selection and classification algorithms. The classifier used for this research was the Bayes classifier using probability density function (PDF) estimates.

This work employs a multivariate kernel density estimator along with the Hellinger Distance to do feature selection. Unlike some other distance measures, the Hellinger does not assume Gaussianity; therefore, the operation requires an estimation of the PDF. Using the Parzen kernel density estimator with a Gaussian kernel, we generated the PDFs for each class. The smoothing parameter for each estimate was selected using an algorithm presented by J. Bibb Cain [1].

Once the PDFs for each training set are generated, the classes are processed with the feature selection algorithms. Each of the three selection algorithms mentioned above were used with the distance measures: Bhattacharyya, Mahalanobis, and Hellinger. The best two and three feature subsets were chosen among the available feature combinations, and the chosen subset for each feature selection algorithm was classified.

Using the Bayes decision theory for classification, we computed the conditional probability densities for each test vector. The ratio of these two probabilities, known as the likelihood ratio $\Lambda(x)$, is computed for each test vector and compared to a threshold value. If the ratio is less than the threshold, the test sample is classified as H_0, and if it is greater or equal to the threshold, it is classified as H_1.

After classification, the test sets are analyzed to determine the correct classification rate. A 95% confidence interval was computed for each correct classification rate and test set size. Another method of evaluating overall performance is the construction of the receiver operating characteristic (ROC) curve. In order to construct the curve, the probabilities of detection and false alarm for each test were computed.

The results for the tests in this research are promising. For the case when both target classes have Gaussian feature distributions, all algorithms performed roughly equally. This is expected because the Bhattacharyya and Mahalanobis distance measures assume the distributions are Gaussian, and as long as the PDF estimate is fairly accurate, the Hellinger distance should provide comparable results.

The disparity in results is first evident when one of the class feature distributions is non-Gaussian. Here, the Gaussian assumption in the Bhattacharyya and Mahalanobis distance measures introduced errors in selecting the proper feature subset. The correct classification rates for these feature selection algorithms was markedly lower than that for the subset chosen by the Hellinger algorithms. For the case when both simulated target classes are non-Gaussian, the Hellinger again outperformed the other distance measures in each of the feature selection algorithms.

In order to establish a benchmark test, the classic Fisher iris data for iris flowers was also used in this research. This data set consists of three classes of irises (Versicolor, Virginica, and Setosa) with each class having four features and 50 measurements of each feature. In order to continue the two-class problem, only the Versicolor and Virginica classes were used. With this data set, the classification rates among the three distance measures was the roughly the same. Despite selecting different subsets of features, the Hellinger and Bhattacharyya algorithms provided a correct classification rate of 95% with two features and 97.50% with three features. The Mahalanobis algorithms provided a 95% classification rate for subsets of two and three features. With all four features, the Bayes classifier achieved a correct classification rate of 95%.

These results demonstrate the idea that feature selection is an important process in the recognition process. Some features obtained in the feature extraction process may be redundant or detrimental to the classification process and can be removed to save computational complexity without a significant loss in classification performance. In order to ensure the proper subset of features is selected, using a distance measure that does not assume a Gaussian distribution proved much more capable in this research. The greater correct classification rates represent an improvement to the existing feature selection algorithms.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ATR	Automatic Target Recognition
B&B	Branch and Bound
KDE	Kernel Density Estimator
LLRC	Log-Likelihood Ratio Classifier
MATLAB	MATLAB program produced by the MathWorks
PDFs	Probability Density Functions
ROC	Receiver Operating Characteristic
SBS	Sequential Backward Selection
SFS	Sequential Forward Selection

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my family for their support, guidance, and inspiration throughout my life. I would also like to thank my wife. Her understanding and tolerance of many late nights completing this research is a testament to her loving nature.

I would also like to thank my advisor, Dr. Grace A. Clark, for her support and mentorship throughout my graduate research. Her willingness to assist whenever needed and her constant words of encouragement and motivation were vital to the completion of this thesis. I would also like to thank Dr. Monique Fargues, the second reader for my thesis. She gave careful and thoughtful feedback and greatly contributed to the quality of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

Automatic target recognition (ATR) is an application of modern pattern recognition. Pattern and target recognition are becoming more commonplace throughout a broad spectrum of applications. Image, voice, facial, and human iris recognition systems are growing in popularity. This scientific discipline involves classifying objects into a number of categories or classes. The objects can be images, waveforms, or any number of measurements that need to be classified. In generic terms, objects are simply referred to as patterns. Most ATR systems can be generally described in terms of the diagram shown in Figure 1.

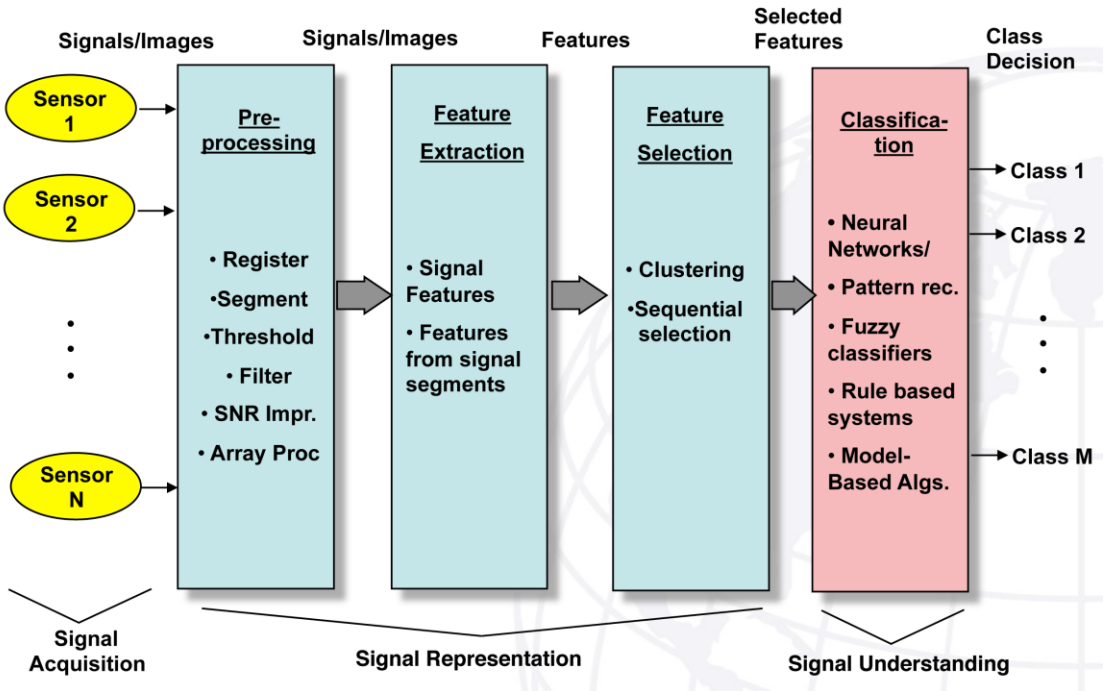

Figure 1. Block diagram for a basic pattern recognition system.

The critical steps in ATR lie in the "signal representation" portion of the system. This area consists of data pre-processing, feature extraction, and feature selection. This area is of great importance because each classifier can only perform as well as the quality of information it is provided. If the classifier is passed "garbage," it will output

"garbage." After the data have been processed and features have been extracted, the feature selection process selects only the most critical subset of features for use in classification. These are the features that provide the maximum separation among the target classes of samples in multi-dimensional feature space. A major problem in pattern recognition is the sheer number of features available for the design of a classifier. The number of features at the disposal of the designer of a classifier can easily reach over a few dozen or even into the hundreds. Reducing the number of features to only a select few greatly reduces the computational complexity of the classification process. Also, feature selection can reduce the effects of mutual correlation between sets of features. Feature selection ensures that added features to the final subset only add significant value to the quality of classification.

There are several techniques available for feature selection. The most commonly used are the Sequential Backward Selection (SBS), Sequential Forward Selection (SFS), Branch and Bound (B&B), and Exhaustive Search [1]. The Exhaustive Search method is the most accurate method as well as the most computationally intensive. This method requires every possible combination of features to be evaluated in order to determine the globally optimal subset of features. For example, if the number of features available is 20 and the designer wishes to limit the feature vector to 10 features, the Exhaustive Search algorithm must evaluate all 184,756 combinations (N things taken P at a time).

Typical criterion functions used in all of these feature selection algorithms often achieve suboptimal results because the functions or algorithms assume the class data are Gaussian distributed when they are not. Traditional feature selection algorithms assume Gaussianity because that case is tractable, not because of optimality. This research aimed to create a new feature selection algorithm that utilized a class separability measure which satisfies the requirements of a metric but does not require the Gaussian assumption. This requires the use of a class separability measure that has not yet been used in feature selection: the Hellinger divergence distance.

B. OBJECTIVE

There are several commonly used class separability measures in feature selection. This research utilized the Bhattacharyya and Mahalanobis distance measures for comparison to the Hellinger distance. The Mahalanobis assumes that the two classes are Gaussian distributed, even if they are not. Also, common implementation of the Bhattacharyya involves restricting the algorithm to the derived expression for the Gaussian case. This invalid assumption can introduce errors into the feature selection algorithm and, ultimately, lead to the selection of a suboptimal feature subset. In an attempt to reduce or eliminate this effect, this research instituted the Hellinger distance measure in three different feature selection algorithms. The Hellinger distance measure does not assume the class distributions to be Gaussian while also satisfying the requirements of metric. This research required data simulation, probability density function (PDF) estimation, feature selection, and classification algorithms, all coded using MATLAB. This research aims to improve the correct classification rate of pattern recognition systems by choosing the best feature subset from the set of all features available by using the Hellinger distance.

C. RELATED WORK

In the areas of pattern recognition and data mining, feature selection is extremely important. Countless research hours have gone into developing, implementing, and evaluating many feature selection algorithms. Mucciardi and Gose compared seven different techniques for choosing subsets of pattern recognition properties [1]. The techniques they evaluated involved correlation tests, eigenvector evaluations, and expected probabilities of error. Their results on simulated and real data show that several feature selection methods can perform equally. Another research project proposed the use of information theory in feature subset selection. Koller and Sahami presented a feature selection algorithm that used the cross-entropy between features to minimize the amount of predictive information lost when discarding features from the subset [2]. Similar research was conducted Yang and Pederson, in which they evaluated information gain, mutual information, and text strength for feature selection [3]. Bissinger used the

Hellinger distance measure in the classification of underwater acoustic signals and compared it to the Log-Likelihood ratio classifier (LLRC) [4]. His results showed that the Hellinger distance was robust to class outlier information and to imperfect class models. The minimum Hellinger distance classifier performed equally or better than the LLRC [4]. These research results all contribute to the motivation behind implementing the Hellinger distance into a feature selection algorithm.

D. APPROACH

For this research, the simulations and codes were written in MATLAB. This required the generation of simulated multivariate data, the estimation of multidimensional PDFs, the evaluation of several divergence measures, and the classification of test data given the subset of features selected from the training data. All MATLAB m-files and functions used in this research are attached in the Appendix.

E. THESIS OVERVIEW AND ORGANIZATION

In Chapter II, the data classification process is explained using the Bayes decision theory as a basis for classification. This includes discussions of the likelihood ratio, correct classification rates, receiver operating characteristic curves, and statistical confidence intervals. The performance measures for a classifier are explained and illustrated thoroughly.

In Chapter III, the theory, mathematics, and motivation behind PDF estimation are presented. The method of PDF estimation used in this research, the Parzen kernel density estimation, is explained. The various PDF estimates that were generated for this research and the selection of the parameters needed for this algorithm are described.

In Chapter IV, the idea of feature selection is introduced. The different feature selection algorithms are then explained in a theoretical sense, followed by each of the three divergence measures evaluated in this research: the Bhattacharyya, Mahalanobis, and Hellinger

In Chapter V, the specific implementation of the research is outlined. Specifically, we describe the simulated data generation, normalization, density

estimation, feature selection, and classification processes applied. Finally, we discuss training and test set issues when applied to classifier problems.

In Chapter VI, we present the results obtained with the various data sets considered. The receiver operating characteristic curves, the correct classification rates, and the confidence intervals for each algorithm are shown. The selected feature subsets are also shown as two-dimensional histograms and scatter plots in an effort to visually show the separability of the classes.

In Chapter VII, we present conclusions and discuss topics for further research.

The Appendix contains the MATLAB m-files and functions used in this research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. CLASSIFICATION

In this chapter, we review the concept of data classification and describe how statistical confidence intervals and receiver operating characteristic curves are used in classification applications.

A. CLASSIFICATION USING BAYES DECISION THEORY

In classification, the statistical variation of the training vectors and noise from the sensors used in feature extraction force decisions to be made based on the probabilities of the data. Adopting this reasoning forces the task of classification to be one of placing a test sample into the most probable class. Given N classes ($\omega_1, \omega_2, \ldots, \omega_M$) and a feature vector $\underline{X}$ corresponding to an unknown class, the N conditional probabilities are expressed as:

$$P(\omega_i \mid \underline{X}), i = 1, 2, \ldots, N. \tag{1}$$

In this research, and in the following section, only the binary hypothesis case (two classes) is analyzed. The problem to be evaluated here is one in which a system generates multiple observations corresponding to one of two hypotheses, H_0 or H_1. Each observation maps to a point in multidimensional feature space, which is the space that corresponds to a set of M observations denoted by the observation vector $\underline{X}$, expressed as:

$$\underline{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}. \tag{2}$$

By definition, observation corresponds to one of the two classes, and the classifier must make the choice between the two. Each time an observation or feature vector is classified, one of four possible events can occur: (1) H_0 is the truth and the feature vector is declared H_0, (2) H_0 is the truth and the feature vector is declared H_1, (3) H_1 is the truth and the feature vector is declared H_1, (4) and H_1 is the truth and the feature vector is declared H_0. The first and third outcomes correspond to correct classification. The second and fourth outcomes correspond to classification errors. The purpose of a

decision criterion is to assign relative importance to the four possibilities. The Bayes test assumes that the prior probabilities (priors) for the hypotheses and the costs associated with the four outcomes are known. The priors, $P(H_0)$ and $P(H_1)$, represent information available to the recognition system prior to any experimentation. The costs for the four possible outcomes are given by C_{00}, C_{10}, C_{11}, and C_{01}, where C_{ij} is the cost of deciding H_i when H_j is the truth. Once the costs have been assigned, the decision rule is based on minimizing the expected cost, known as the Bayes risk $\Re$, and shown as [5]:

$$\Re = \sum_{i=0}^{1}\sum_{j=0}^{1} C_{ij} P(H_i \mid H_j) P(H_j). \tag{3}$$

It is assumed throughout this section that the cost of an incorrect decision is higher than the cost of a correct decision. In terms of C, $C_{10} > C_{00}$ and $C_{01} > C_{11}$. Given this assumption, the detector that minimizes the Bayes risk is given by:

$$\frac{f(\underline{X} \mid H_1)}{f(\underline{X} \mid H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \frac{P(H_0)(C_{10} - C_{00})}{P(H_1)(C_{01} - C_{11})}, \tag{4}$$

where the ratio of the conditional probabilities on the left-hand side of the equation is called the likelihood ratio. The likelihood ratio is denoted by $\Lambda(x)$ and is expressed as:

$$\Lambda(x) = \frac{f(\underline{X} \mid H_1)}{f(\underline{X} \mid H_0)}. \tag{5}$$

A very important result of the likelihood ratio is that, regardless of the dimensionality of the observation $\underline{X}$, the ratio is a scalar value. This idea is critical in hypothesis testing. This means that regardless of the observation feature space dimension, the decision space is one-dimensional. The right-hand side of the inequality above is the test threshold and is denoted as:

$$\eta \triangleq \frac{P(H_0)(C_{10} - C_{00})}{P(H_1)(C_{01} - C_{11})}, \tag{6}$$

where η is the threshold value. The previous assumptions and equations present the Bayes criterion as a likelihood ratio test, shown as:

$$\Lambda(\underline{X}) \underset{H_0}{\overset{H_1}{\gtrless}} \eta. \tag{7}$$

Since the threshold value depends on the priors and the weights assigned to the costs, the designer has flexibility in choosing a threshold that is best for the recognition problem at hand. The next step in the design is to evaluate the classifier performance using the probability of correct classification P_{CC}. For this research, it is assumed that a correct classification is assigned zero cost (C_{00}= C_{11}=0) and an incorrect classification is assigned full cost (C_{01}= C_{10}=1). With this assumption, and understanding that the sum of the probability of error P_E and the P_{CC} is one, the probability of correct classification is defined as:

$$P_{CC} = P(H_1 \mid H_1)P(H_1) + P(H_0 \mid H_0)P(H_0). \tag{8}$$

The probability of correct classification is a very important performance measure. In this research, it is assumed that the two classes have an equal probability of occurrence, so that $P(H_0) = P(H_1) = 0.5$. The previous assumptions are common in pattern recognition because there is often an insufficient amount of information about an experiment to allow assignment of priors and costs. From these assumptions, the probability of correct classification becomes:

$$P_{CC} = \frac{1}{2}\left[P(H_1 \mid H_1) + P(H_0 \mid H_0)\right]. \tag{9}$$

Another method of evaluation for the classifier is to determine the probabilities of detection P_D and false alarm P_{FA}. These probabilities are defined in the confusion matrix or contingency table shown in Figure 2.

"Confusion Matrix" or Contingency Table for Binary Hypothesis Testing		
Truth / Declaration	True Hypothesis H_0 (Null)	True Hypothesis H_1
Declared Hypothesis H_0(Null)	$P(H_0 \mid H_0) = P_{Spec} = Specificity$ $= \frac{\#H_0\ Samples\ Declared\ H_0}{\#H_0\ Samples}$	$P(H_0 \mid H_1) = P_{Miss} = P(Miss)$ $= \frac{\#H_1\ Samples\ Declared\ H_0}{\#H_1\ Samples}$
Declared Hypothesis H_1	$P(H_1 \mid H_0) = P_{FA} = P(False\ Alarm)$ $= \frac{\#H_0\ Samples\ Declared\ H_1}{\#H_0\ Samples}$	$P(H_1 \mid H_1) = P_D = P(Detection)$ $= \frac{\#H_1\ Samples\ Declared\ H_1}{\#H_1\ Samples}$

Figure 2. Confusion Matrix for a two-class classification problem. From [6].

From the confusion matrix, the probability of correct classification can now be expressed in terms of the probability of detection and the probability of false alarm, shown as:

$$P_{CC} = \frac{1}{2}\left[P_D + (1 - P_{FA})\right]. \tag{10}$$

Figure 3 is an example of a two-class classification problem. The red line displayed is the chosen threshold, η.

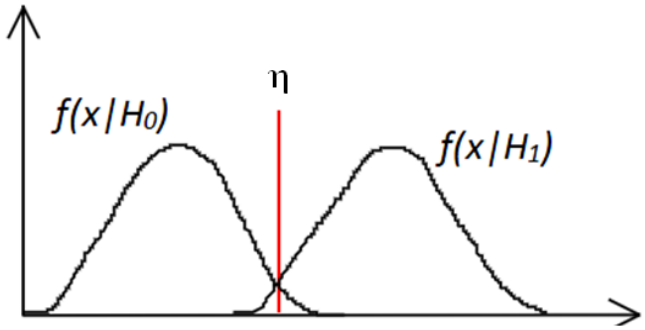

Figure 3. A two-class PDF classification problem for the one-dimensional feature vector case.

The portion of the conditional probability $f(x|H_0)$ to the left of the threshold is classified as H_0. The portion of the conditional probability $f(x|H_1)$ to the right of the threshold is classified as H_1. There is a small portion of the $f(x|H_0)$ PDF that is to the right of the threshold. This area would be incorrectly classified as H_1. From the confusion matrix, this error is a false alarm. Conversely, the portion of the $f(x|H_1)$ PDF to the left of the threshold would be incorrectly classified, known as the miss error from the confusion matrix.

B. STATISTICAL CONFIDENCE INTERVAL ON THE PROBABILITY OF CORRECT CLASSIFICATION

After estimating the P_{CC}, it is important to estimate the confidence with which the system can specify the performance of the classifier. In order to do this, the confidence interval for the probability of correct classification is estimated [7]. In the process of evaluating the classifier's performance, the conditional probabilities are estimated based

upon experiments with the data. The classifier declares each feature vector as belonging to one of two classes, giving either a success or failure. Success simply means the classification was correct, while failure means it was incorrect. If N equals the number of test feature vectors and $\hat{p}$ denotes the probability of success ($\hat{p} = P_{CC}$), the maximum likelihood estimate of the probability of success is expressed as:

$$\hat{p} = \frac{\text{Number of Correct Classifications}}{N}. \tag{11}$$

The confidence interval about the true value of P_{CC} is expressed as:

$$P(L < P_{CC} < U) = 1 - \alpha \tag{12}$$

where α is the significance of the test and L and U are the lower and upper bounds, respectively, of the confidence interval. Equation 12 simply reads that with confidence $(1-\alpha)$, the true P_{CC} lies between the lower and upper bounds. In this research, α was set to 0.05, providing a 95% confidence interval. In order to determine the lower and upper bounds, the binominal approximation to the distribution of P_{CC} was used because it offers more accurate results in comparison to the Gaussian assumption [8]. The estimates of the bounds are given by [9]:

$$L = \frac{N\hat{p} + 2 - 2\sqrt{N\hat{p}(1-\hat{p}) + 1}}{N + 4} \tag{13}$$

and

$$U = \frac{N\hat{p} + 2 + 2\sqrt{N\hat{p}(1-\hat{p}) + 1}}{N + 4}. \tag{14}$$

These expressions demonstrate the importance of having a large amount of test data. As the number of test vectors increases, the confidence interval becomes smaller, providing more accurate results. In Figure 4, the shrinking of the interval as N is increased is demonstrated [6].

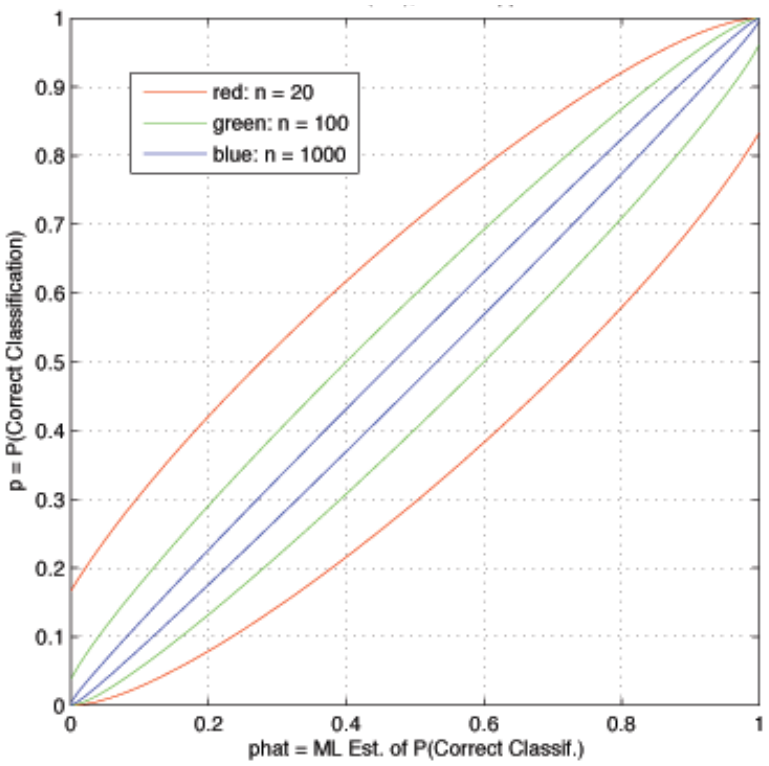

Figure 4. 95% confidence interval for various data set sizes. The 'x' axis corresponds to the maximum correct classification rate and the 'y' axis corresponds to the 95% confidence interval bounds.

C. RECEIVER OPERATING CHARACTERISTIC CURVE

From the previous discussion concerning classification, the next step in evaluating the classifier's performance is to look at its receiver operating characteristic (ROC) curve.

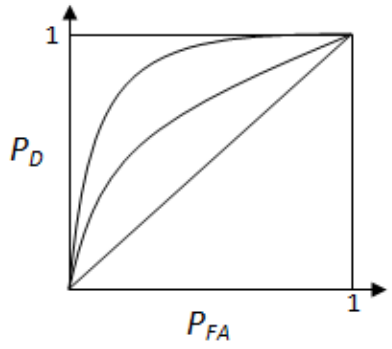

Figure 5. Example of a family of ROC curves. Curves bend towards the upper left corner as the class separation increases.

When the two class PDFs overlap exactly, then the P_D and P_{FA} are equal over a range of thresholds, corresponding to the 45° straight line in Figure 5. As the two distributions move apart, the ROC curve over the range of thresholds shifts away from the straight line. This shifting of the ROC curve is the desired effect. The larger the area between the curve and the straight line, the better the classifier performs. The ROC curve generation is another vital evaluation of the classifier, and is used in this research to show the performance of the feature selection algorithms with the various class separability measures.

In this chapter, the classification process was thoroughly described, which included a discussion of the Bayes decision theory and the concepts of statistical confidence intervals and the receiver operating characteristic curve. In the next chapter, the PDF estimation process is explained.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DENSITY ESTIMATION

The histogram computed for data samples provides the simplest approach to estimate a probability density function *f(x)*. However, while it is effective, the histogram approach does not provide the best estimate of the probability density function. In this chapter, we present a kernel density estimator approach designed to provide a more accurate estimate of the PDF than that obtained with the histogram.

A. PARZEN KERNEL PDF ESTIMATION

The kernel density estimator (KDE) PDF estimate is much more sophisticated than the histogram estimate. The Parzen kernel estimator was used in this research because of its generality, ease of use, and robustness as demonstrated by a wide variety of applications in [10]–[13]. The Parzen kernel PDF estimator is the basis for the Probabilistic Neural Network [14]. The PNN is actually a Bayes optimal classifier, which uses PDF estimates based upon the Parzen kernel. The most commonly used kernel is the normal or Gaussian kernel. The Parzen Gaussian kernel PDF estimator is given by:

$$f(\underline{X}) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^{p}} \sum_{i=1}^{M} \exp\left[\frac{-(\underline{X}-\underline{X}_i)^{T}(\underline{X}-\underline{X}_i)}{2\sigma^{2}}\right], \tag{15}$$

where the parameters are:

i = Training vector index for vector under test over the range [1, M],

M =Number of training vectors in the training set,

$\underline{X}$ = Training feature vector ($\underline{X} = [x_1, x_2, .., x_p]^{T}$),

σ = Smoothing parameter for the PDF estimator (chosen by Cain algorithm described below),

p = Dimension of the feature vector $\underline{X}_i$ from the training set.

From Equation 15, both training set PDF estimates can be generated, one corresponding to each class.

The width of the Gaussian kernel is specified by the smoothing parameter σ, which is the standard deviation of the kernel. The PDF estimate is simply the summation of all the kernels over the entire training data set consisting of *M* samples. The smoothing parameter choice directly affects the quality of the PDF estimate. When the smoothing parameter is too small, the PDF estimate contains distinct modes and appears rough. Conversely, when the smoothing parameter is selected to be too large, the PDF estimate is unable to capture fine structure.

1. Density Estimation for Test Sets

In order to properly estimate the conditional probabilities for each test vector for classification purposes, the vector must be evaluated using the KDE previously explained. The training set PDF estimates are generated over a grid of values while the test set conditional probability estimates are generated for each test vector point. The conditional probability for a test vector $\underline{X}_i$ given it is of class H_0 is expressed as:

$$f(\underline{X}_i \mid H_0) = \frac{1}{(2\pi)^{\frac{p}{2}}\sigma^p} \sum_{j=1}^{N_{H0}} \exp\left[\frac{-(\underline{X}_i - \underline{X}_{H0,j})^T(\underline{X}_i - \underline{X}_{H0,j})}{2\sigma^2}\right], \tag{16}$$

where the parameters are:

i = Test vector index for vector under test over the range [1, M],

j = Training vector index over the range [1, N_{H0}],

M =Number of test vectors in the test set,

N_{H0} = Number of training vectors in the training set H_0,

$\underline{X}_i$ = Test feature vector ($\underline{X} = [x_1, x_2, .., x_p]^T$),

$\underline{X}_{H0,j}$ = Training feature vectors for class H_0,

σ = Smoothing parameter for the PDF estimator (chosen by Cain algorithm described below),

p = Dimension of the feature vector $\underline{X}_i$ from the training set.

The same process is used to generate the conditional probabilities for each test point given it is of class H_1. These conditional values are then used in the likelihood ratio.

B. SMOOTHING PARAMETER SELECTION

Optimum smoothing parameter selection is typically a task requiring trial and error with multiple refinements and lots of time. Despite this drawback, several algorithms and techniques exist for selecting the parameter automatically. This research used the approach proposed by J. B. Cain, which provides a reasonable estimate of the smoothing parameter for a given training dataset [15]. The Cain algorithm was originally intended for use with the PNN, which is a Bayes classifier, utilizing the PDF estimates from a Parzen PDF estimator. Cain's algorithm is based upon the observation that the PDF estimate at any point should be significantly influenced by more than one training vector but not by a large number of vectors. It is clear that the larger the number of training vectors used and the more densely distributed these vectors are, the smaller the smoothing parameter must be for quality performance in the PDF estimation. In this algorithm, σ is proportional to the average distance between the training vectors within the same class. If i denotes the training vector index, p_i denotes the i-th training vector, and $|C_k|$ denotes the number of training vectors within the class, the distance between training vectors d_i is expressed as:

$$\hat{d}_{avg} = \frac{1}{|C_k|} \sum_{p_i} d_i. \tag{17}$$

Finally, the smoothing parameter is assigned to be:

$$\sigma = g\hat{d}_{avg}, \tag{18}$$

where g is in the range ($1.1 \leq g \leq 1.4$), which was determined empirically by Cain and found to be useful in a variety of applications [15].

This chapter presented the kernel density estimator designed to estimate the PDF of discrete samples used in this work. In the next chapter, the feature selection methods used in this work are thoroughly explained.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. FEATURE SELECTION

In this chapter, we introduce the concepts of feature selection and the three commonly used algorithms considered in the research: the exhaustive search, the branch and bound method, and the sequential forward selection method. Next, we explain the three divergence measures considered in this research: Bhattacharyya, Mahalanobis, and Hellinger distance measures.

A. FEATURE SELECTION ALGORITHMS

In an effort to mitigate the "curse of dimensionality," many pattern recognition systems rely on feature selection to reduce the amount of information required to define classes and to be used to classify new sets of data. This research evaluates the performance of three common feature selection algorithms with the three class separability measures presented.

1. Exhaustive Search

The Exhaustive Search algorithm is a feature selection method in which all possible combinations of the desired size feature vector are enumerated, evaluated, and compared. The combination that provides the greatest amount of class separability is the chosen feature subset vector. This method is by far the most computationally intensive selection algorithm. Given the number of available features n and the number of features to be used in the subset k the number of combinations to be evaluated is expressed as:

$$\text{No. of Combinations} = \binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{19}$$

2. Branch and Bound

The Branch and Bound (B&B) algorithm is a much faster alternative to the Exhaustive Search and can also provide the globally optimal feature subset without exploring every possible combination when used in conjunction with a monotonic criterion function. The worst case B&B scenario is that it may require the same number of computations as the Exhaustive Search does, but this case is very rare. B&B avoids

evaluating all feature combinations by rejecting suboptimal subsets without direct evaluation and guarantees that the selected subset yields the globally optimal solution when the selection criterion satisfies the monotonicity condition:

$$J_1(x_1) \le J_2(x_1, x_2) \le \ldots J_m(x_1, x_2, ..., x_m), \tag{20}$$

where $J_i(x_1, x_2, ..., x_i)$ is the criterion function evaluated for all features $x_1, x_2, ..., x_i$ from the feature set. This restriction simply implies the resultant criterion value should increase as the number of features used increases (i.e., is monotonically increasing). The Branch and Bound algorithm used in this research can be found in [16]. In this algorithm, the number of available features is defined as *n*. The goal is to select the best subset of *m* features so that the class separability measure is optimized over all subsets of size *m*. The algorithm starts with the full set of features, and as it branches down each level, a feature is discarded, as shown in Figure 6 [16]. The criterion function is evaluated at each node and compared to the current bound value. The first bound value is the value of the criterion function for the node at the bottom-right side of the solution tree. After this bound is established, if a node higher in the solution tree provides a distance measure less than the bound, the solutions stemming from that node do not require evaluation and can be disregarded. This characteristic of avoiding evaluation of all possibilities is what makes B&B a much more feasible approach as compared to the Exhaustive Search.

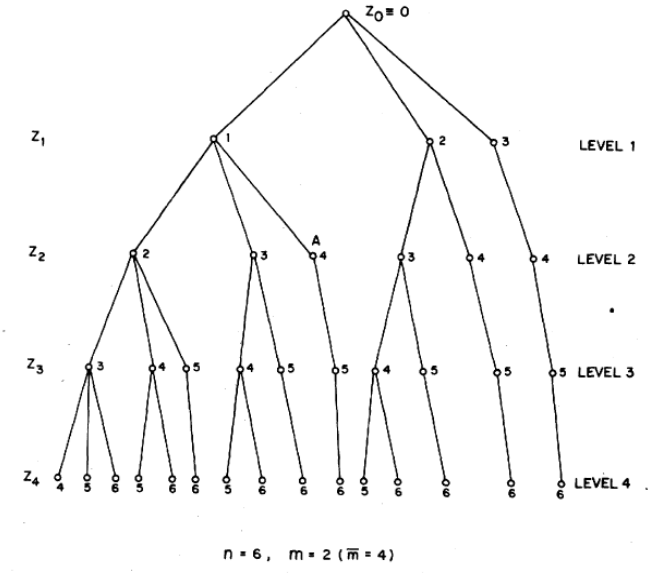

Figure 6. An example of a solution tree for the best two features for the Branch and Bound algorithm with six available features. From [16].

3. Sequential Forward Selection

The Sequential Forward Selection (SFS) algorithm requires that each feature be ranked according to its contribution to class separability. SFS uses a bottom-up search strategy to obtain the final feature subset. The same ranking scheme is used for Sequential Backward Selection (SBS), but the SBS algorithm uses a top-down approach. Features are either added to or removed from the current set based on the measured increase or decrease in class separability. These algorithms cannot guarantee optimality because the selection or discarding of features is done on a scalar basis, meaning each feature is evaluated individually. The best overall subset combination does not necessarily contain the top individual features of a data set. The SFS uses a bottom-up search strategy, in which the starting set is the null set of features. One feature at a time is included in the current feature set.

For SFS, we first specify a priori the number of features *b* desired in the final feature set. At each stage in the algorithm, one feature is added to the current feature set.

The new feature is selected from the set of features not already in the current feature set. For a new feature to be included, the new enlarged feature set must yield the maximum value of the criterion function.

B. CLASS SEPARABILITY MEASURES

For a distance measure to be employed in a feature selection algorithm, it is desirable that the measure possess the following four mathematical properties of a metric. If $d[f(x), g(x)]$ denotes the distance between two PDFs *f(x)* and *g(x)*, the four properties are:

(1) **Identity:** $d[f(x), g(x)] = 0$ if *f(x)=g(x)*. The distance between two identical objects should be zero.

(2) **Non-Negativity:** $d[f(x), g(x)] \geq 0$. To conform to traditional concepts of distance, the distance should be non-negative.

(3) **Symmetry:** $d[f(x), g(x)] = d[g(x), f(x)]$. The distance between *f(x)* and *g(x)* is the same as the distance from *g(x)* to *f(x)*.

(4) **Triangle Inequality:** $d[f(x), h(x)] \leq d[f(x), g(x)] + d[g(x), h(x)]$.

A distance should obey the property that the distance between *f(x)* and *g(x)* plus the distance between *g(x)* and *h(x)* should be less than or equal to the distance between *f(x)* and *h(x)*. This allows the distances among objects to be compared easily and reinforces the traditional concept of distance.

1. Divergence

Many class separability measures are forms of a divergence distance measure. These criteria measure the separation between two probability density functions *f(x)* and *g(x)*. Various divergence measures are defined in terms of the likelihood ratio of the two densities, such as:

$$\lambda = \frac{g(x)}{f(x)}. \tag{21}$$

The Mahalanobis distance measure is a divergence measure that assumes that both class PDFs are Gaussian distributed. Common implementation of the Bhattacharyya distance utilizes the Gaussian assumption in an effort to avoid PDF estimation. The Hellinger distance is also a divergence measure but does not require the Gaussian assumption.

a. The Bhattacharyya Distance

In terms of the associated class PDFs, the Bhattacharyya distance is expressed as:

$$B(f,g) = \int_x \sqrt{f(x)g(x)}dx\,. \tag{22}$$

where *f(x)* and *g(x)* are the two class PDFs and B is the Bhattacharyya distance. The Bhattacharyya distance measure for Gaussian data is expressed as:

$$B = \frac{1}{8}(\mu_i - \mu_j)^T \left(\frac{\sum_i + \sum_j}{2}\right)^{-1}(\mu_i - \mu_j) + \frac{1}{2}\ln\left[\frac{|(\sum_i + \sum_j)/2|}{|\sum_i|^{1/2}|\sum_j|^{1/2}}\right], \tag{23}$$

where *i* and *j* represent the two different classes, μ is the corresponding class mean vector, Σ is the corresponding covariance matrix, and *B* is the Bhattacharyya distance

The Bhattacharyya distance satisfies the identity, non-negativity, and symmetry properties of a distance measure, but it does not obey the triangle inequality. The relationship between Equations 21 and 23 is explored in [17].

b. The Mahalanobis Distance

The Mahalanobis distance measure is very similar to the Bhattacharyya distance. The Mahalanobis distance for Gaussian data is expressed as:

$$M = (\mu_i - \mu_j)^T \Sigma^{-1}(\mu_i - \mu_j) \tag{24}$$

where *i* and *j* again represent the two different classes, μ is the corresponding class mean vector, Σ is the average of the two class covariance matrices, and *M* is the Mahalanobis distance.

c. The Hellinger Distance

The squared Hellinger distance is defined as [4]:

$$H^2(f,g) = \frac{1}{2}\int_x [\sqrt{f(x)} - \sqrt{g(x)}]^2 dx, \tag{25}$$

where *f(x)* and *g(x)* are the two class PDFs and H is the Hellinger distance. Moving the square root to the right-hand side of the equation, we get the Hellinger distance as:

$$H(f,g) = \frac{1}{\sqrt{2}}\sqrt{\int_x [\sqrt{f(x)} - \sqrt{g(x)}]^2 dx}. \tag{26}$$

The Hellinger distance satisfies all four properties of a metric, and its range is [0,1]. This makes it an ideal candidate for use in estimation, feature selection, and classification. The robustness of minimum Hellinger distance methods has been explored in [4]. In order to use the Hellinger distance for feature selection, it must be written for discrete variables and in vector form since the feature selection algorithms require operation on multidimensional PDFs. The scalar Hellinger distance for discrete variables is given by:

$$H(f,g) = \frac{1}{2}\sqrt{\sum_{k=0}^{K-1} [\sqrt{f(x_k)} - \sqrt{g(x_k)}]^2 dx}\,, \tag{27}$$

where k is the discrete index over the range k=[0, K-1] and K is the integer number of measured samples of x_k. In order to compute the Hellinger distance between two multivariate PDFs, this equation must be extended to the vector or multidimensional case. Here, the PDFs are shown as $f(\underline{X})$ and $g(\underline{X})$. The vector $\underline{X}$ is now an N by 1 feature vector containing N features. There are M measured training feature vectors such that $\underline{X}_m$, m= (1,M). With this notion, the multivariate Hellinger distance is given by:

$$H(f,g) = \frac{1}{2}\sqrt{\sum_{m=1}^{M} [\sqrt{f(\underline{X}_m)} - \sqrt{g(\underline{X}_m)}]^2 d\underline{X}}\,. \tag{28}$$

The concepts of feature selection and class separability measures were reviewed in this chapter. The specific feature selection algorithms and class separability measures used in this research were also described. In the next chapter, the specific

implementation of this research is outlined. This includes discussions of the data generation, PDF estimation, normalization, feature selection, and classification processes used in this research.

THIS PAGE INTENTIONALLY LEFT BLANK

V. EXPERIMENTAL SETUP

Experiments and tests conducted in this research required the generation of multivariate data sets, multidimensional PDF estimation, coding of feature selection algorithms and distance measures, and the classification and evaluation of the data and algorithms used. All simulations were conducted using MATLAB and are described in this chapter.

A. DATA SIMULATION

The first step in evaluating the Hellinger distance feature selection algorithms was to generate multivariate data to be used as target class data for H_0 and H_1. The observation data were generated using a normally distributed multivariate number generator in MATLAB. The generators require the mean vectors, covariance matrices, and number of points for each data set to be generated. For this research, the means and covariance matrix values were selected randomly over the range [0, 14]. Covariance matrices were constructed as diagonal matrices to ensure they were positive semi-definite. Finally, two Gaussian distributions were combined to generate a non-Gaussian distribution, as depicted in Figure 7.

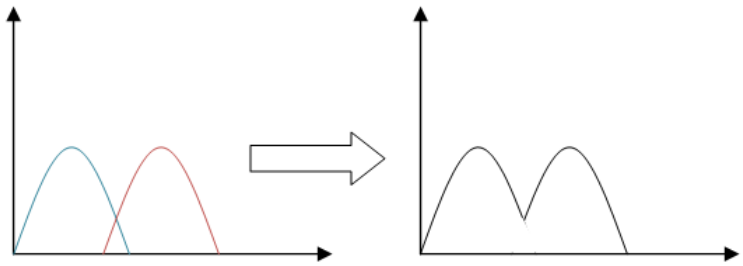

Figure 7. Combination of two one-dimensional Gaussian distributions to provide a non-Gaussian probability density.

As a result of creating non-Gaussian data in this manner, the features in a class may become correlated. This result does not affect the performance of any algorithm.

The structure for the observed data is the same regardless of distribution. Each row represents a measured feature vector containing N features, meaning each column

represents a measured value for each feature. For example, if the data set generated is of size 1000 by 8, this means there are 1000 different feature vectors each containing eight features. The structure for the observed data generated is shown in Figure 8.

$$Class = \begin{bmatrix} & Feature1 & Feature2 & \cdots & FeatureN \\ Measurement1 & X_{11} & X_{12} & X_{1\ldots} & X_{1N} \\ Measurement2 & X_{21} & X_{22} & X_{2\ldots} & X_{2N} \\ \cdots & X_{\ldots 1} & X_{\ldots 2} & X_{\ldots\ldots} & X_{\ldots N} \\ MeasurementM & X_{M1} & X_{M2} & X_{M\ldots} & X_{MN} \end{bmatrix}$$

Figure 8. The data structure for each class generated using the multivariate distribution generator in MATLAB is shown.

The two simulated data sets for the two classes are then used to evaluate the performance of the various feature selection algorithms. Before any PDF estimates or distances are computed, the class data were first normalized on the interval [−1, 1]. The data were normalized by feature for both classes. For example, looking at Figure 8, the "Feature 1" column would be normalized based on that individual column's minimum and maximum values. The same is done for all *N* columns of both classes. Normalizing the data ensures that features with larger values do not dominate the selection algorithms and also simplifies the grid building process for the density estimation.

B. DENSITY ESTIMATION

The Bhattacharyya and Mahalanobis distance measures both assume that the classes are Gaussian distributed, removing the need to compute an estimate of the PDF. Since the Hellinger distance does not assume Gaussianity, it requires an estimate of the PDF to compute the distance. As discussed in Chapter II, the PDF estimation method used in this research is the Parzen kernel density estimator.

First, the training data sets are reshaped to include only the features in each class being evaluated. For example, if a PDF estimate is needed for features two and three of each class, the other feature columns are stripped away, leaving only an *M* by 2 feature matrix for each class.

The next step requires determining a feature grid over which to compute the PDF estimates. This is done to ensure that, over the given range of the data, an estimate of the PDF is computed for each point in the grid, not just for each observation of the feature matrix. The training data PDF estimates are computed over the same grid of values to ensure each point in both estimates has a value to allow for evaluation of the Hellinger distance. Such a task requires the user to specify a grid scale factor. The grid scale factor is used to extend the PDF estimate beyond the minimum and maximum values of the data set in an effort to estimate the "tails" of the PDF. An example illustrating this step is demonstrated in Figure 9.

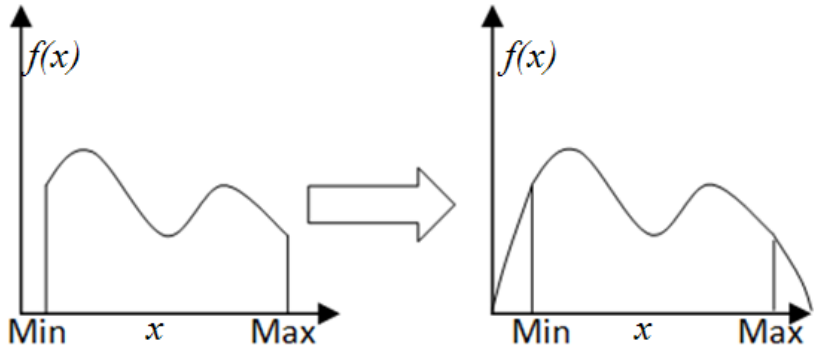

Figure 9. Use of the grid scale factor in estimating the PDF tails for the one-dimensional case.

The user is also required to specify the sampling interval *dX* along each dimension of the PDF estimate. This value, along with the grid scale factor and the minimum and maximum values of the data, are used to determine the grid of points to be used in the PDF estimate. For example, after normalization, the class minimum and maximum values are −1 and 1, respectively. If the grid scale factor is specified as 1.5 and the sampling interval is set to 0.1, the grid range is [−1.5, 1.5] and has points evenly spaced by 0.1 throughout the range.

The last step in computing the PDF estimate is to pass the proper reshaped training sets to a function that computes the PDF estimate for each point on the grid given

the training set for that specific class. This function requires the use of a smoothing parameter σ discussed in Chapter II. The result from this function is an estimate of the class PDF given a specific feature subset.

C. FEATURE SELECTION ALGORITHMS

The various feature selection algorithms to be evaluated are the Exhaustive Search, Branch and Bound, and Sequential Forward Selection. The details and explanations for each of these algorithms are given in Chapter IV. Each feature selection algorithm is used with each of the three class separability measures (Mahalanobis, Bhattacharyya, and Hellinger) for comparison purposes. Each algorithm requires the user to specify the desired number of features in the final feature vector subset. The output for each of these functions is simply the best combination of features from the available set.

After the algorithm has determined the best subset of features, the training and test sets must be transformed into "final" training and test sets. This requires each feature that is not selected during feature selection to be discarded from every training and test vector for each class. For example, if eight features were available and the feature selection algorithm selected feature columns two and four, columns one, three, five, six, seven, and eight would be removed from the sets, creating *M* by 2 matrices for each class of training and test vectors. The last step is to add two columns to the test data. The next to last column is the truth column, which contains information about the class to which the vector truly belongs (one for H_0 and two for H_1). The last column is left blank and is used to hold the declaration or classification decision. These matrices are then passed to the Bayesian classifier for evaluation.

D. CLASSIFICATION OF TEST DATA

Once the final training and test sets have been created using only the selected subset of features, the test data can be classified. The training set for each class, along with each test vector, are passed to the kernel density estimator (KDE) function to estimate the conditional probability density for each class. These scalar values are then used to determine the likelihood ratio for each test feature vector. As discussed in Chapter II, these ratios are then compared to a range of threshold values in order to

construct the ROC curve and to determine the optimum threshold. The optimum threshold is that which provides the maximum correct classification rate. The likelihood ratio values are compared to the threshold values, and a decision is made based on the inequality expression. The decision is placed into the last column of the test data. Once every test vector has been classified, the decisions are compared to the truth. From the previous example of two features in a final subset, features two and four, a possible test matrix output from the classifier is shown in Figure 10.

$$Test = \begin{bmatrix} & Feature1 & Feature2 & Truth & Declaration \\ Vector1 & X_{11} & X_{12} & 1 & 1 \\ Vector2 & X_{21} & X_{22} & 1 & 1 \\ \cdots & X_{...1} & X_{...2} & 2 & 2 \\ VectorM & X_{M1} & X_{M2} & 2 & 1 \end{bmatrix}$$

Figure 10. Output test matrix example from the Bayes classifier.

In this example, the classifier made one error, and the correct classification rate is 4/5 or 80%. The statistical confidence interval is then computed with the corresponding correct classification rate and the size of the test data.

The setup and approach to this research experiment were explained in this chapter. The data generation, density estimation, feature selection, and classification processes used specifically in this research were also explained. In the next chapter, the results from the various test runs are presented and explained.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. RESULTS

In order to test the algorithms presented in this thesis, several simulated datasets were generated and used. The results from these algorithms are presented in this chapter, along with supporting details and conclusions.

A. SIMULATED DATA

1. Two Gaussian Distributed Target Classes

The first test of the feature selection algorithm with two Gaussian target classes involved both classes having a combined 1,400 training vectors (700 per class) and 1,000 test vectors (500 per class) with a total of eight available features. The best two features were then selected among the available combinations of features. The algorithm evaluated for this case was the exhaustive search method. This algorithm was implemented three times on the set of data, once for each class separability measure discussed. The distributions of each of the eight features in feature space for each of the two classes are shown in Figures 11 and 12. The distributions were calculated after each generated data set was normalized as discussed in Chapter V.

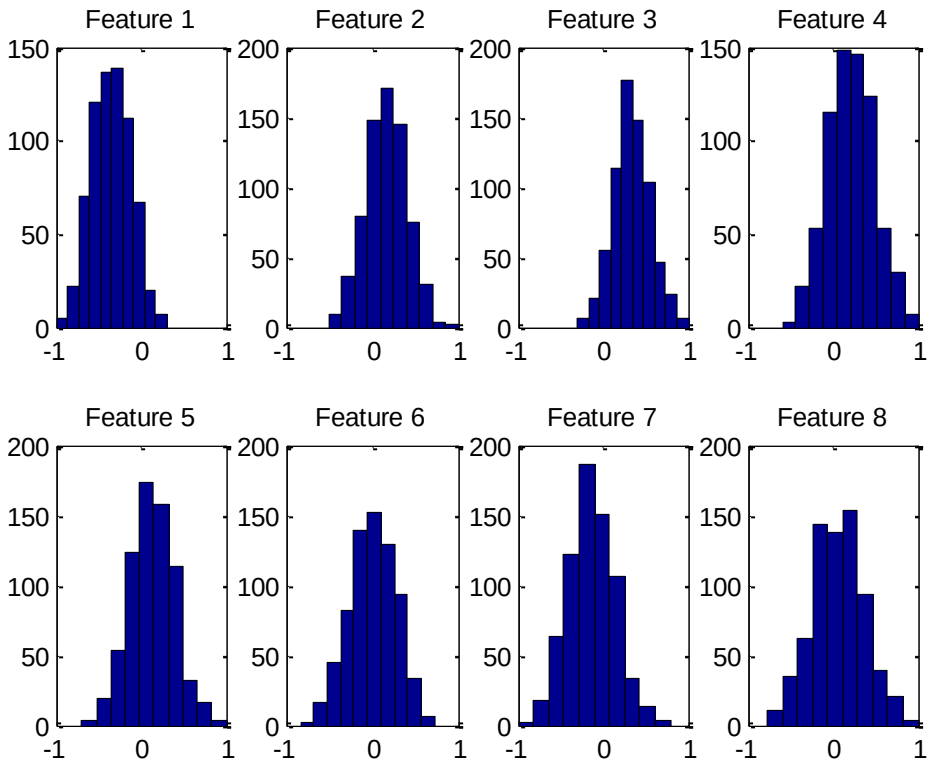

Figure 11. Feature distributions for each feature of H_0 in feature space for a Gaussian distributed target class.

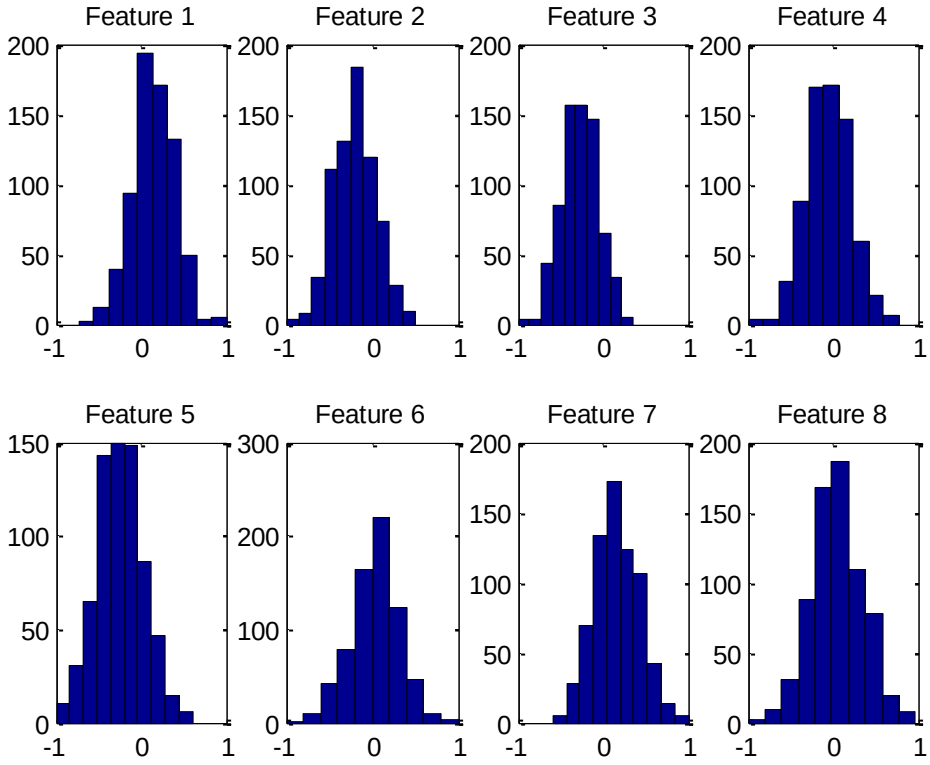

Figure 12. Feature distributions for each feature of H_1 in feature space for a Gaussian distributed target class.

The first class separability measure used in the exhaustive search method was the Hellinger distance measure. Since the Hellinger distance method requires the estimation of the class PDFs, the algorithm was given a Grid Scale Factor of 1.1 and a sampling interval *dX* of 0.1. The grid scale factor was varied over a range of [1.1, 2.0] and did not affect the algorithm's results. Similarly, the sampling interval was varied between 0.0001 and 0.1 and this also did not affect the results. With the data set described above, the Hellinger Exhaustive Search algorithm selected features three and five. This information was then used to construct the final training and test vectors as discussed in the Chapter V. The two-dimensional histograms for each training set with only the selected features are shown in Figure 13.

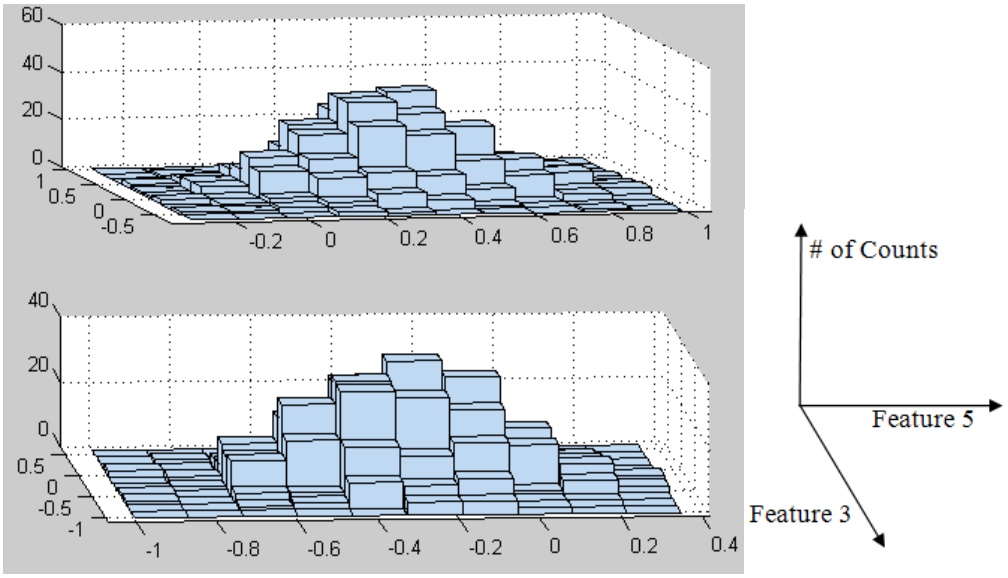

Figure 13. Two-dimensional histograms of the two classes with the selected subset of features, three and five, obtained using the exhaustive search with the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 13 is shown as a scatter plot in Figure 14. These two figures show the separability of the two class PDFs chosen by the Hellinger algorithm.

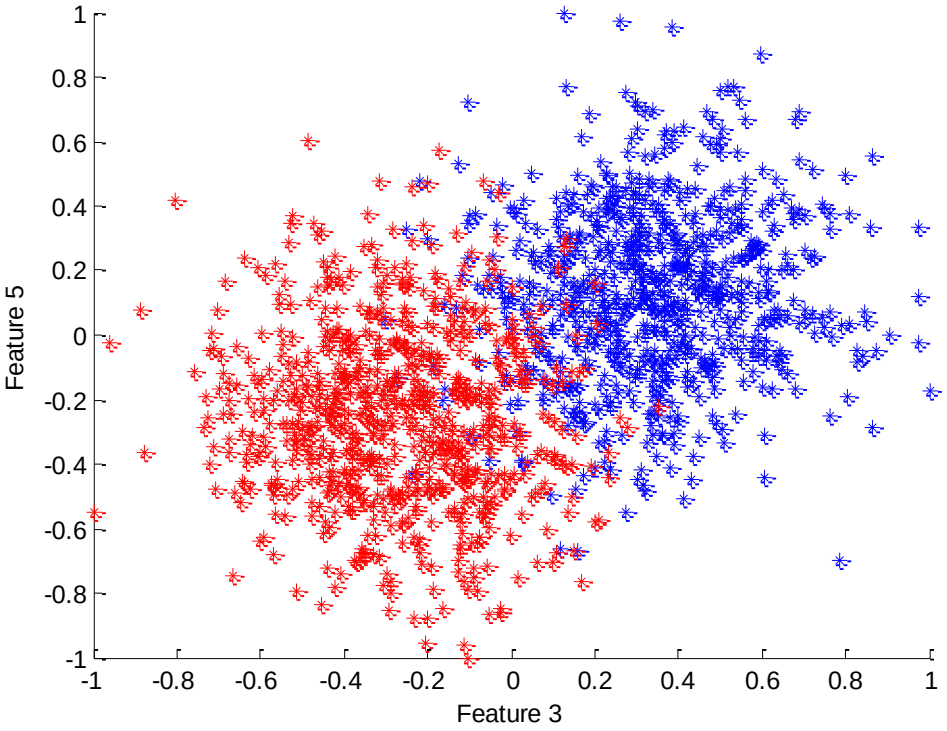

Figure 14. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 94.00% of the test data. The ROC curve for this implementation is shown in Figure 15.

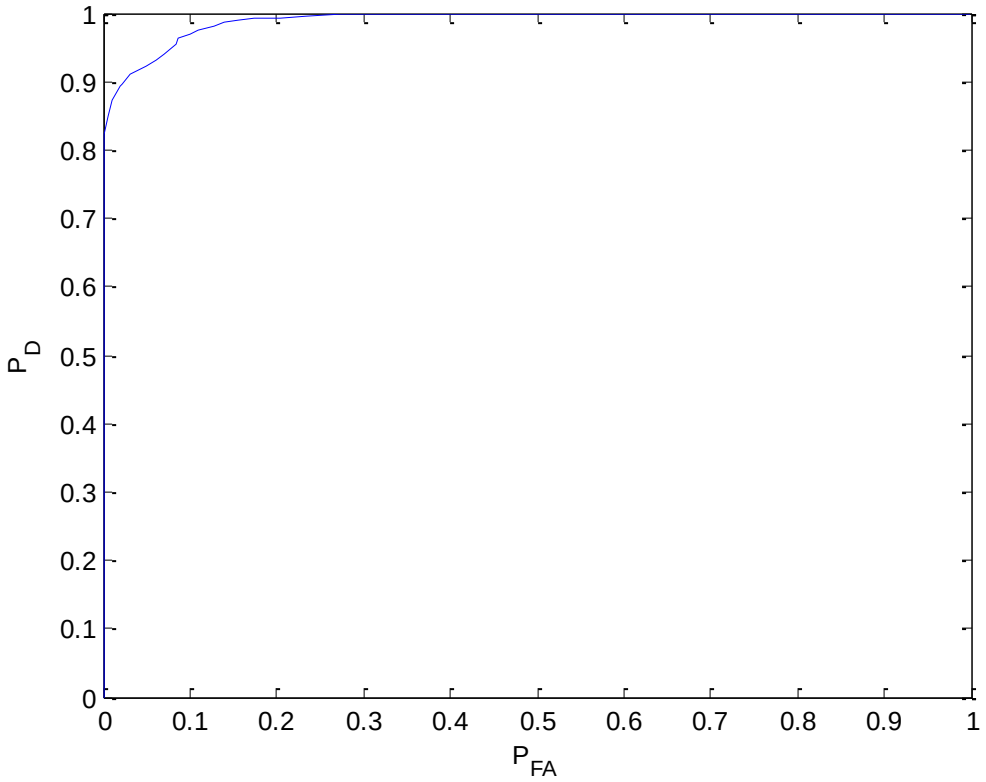

Figure 15. ROC curve obtained with the exhaustive search method and the Hellinger distance for the two Gaussian distributed target classes using features three and five.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 16.

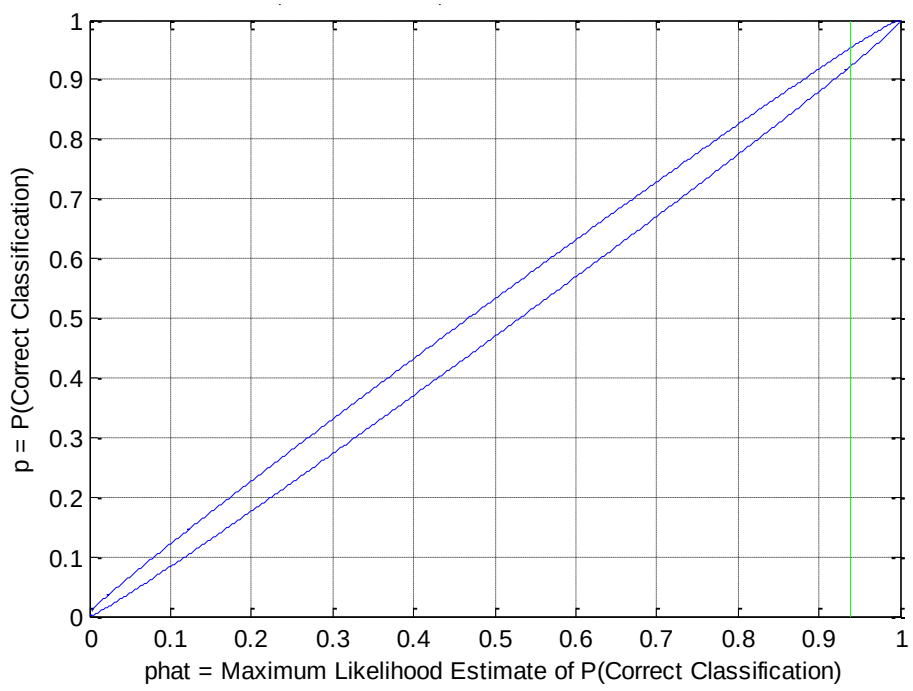

Figure 16. 95% confidence interval for the correct classification rate of 94.20% for the exhaustive search using the Hellinger distance results with a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 92.31% and an upper bound of 95.44%. The same data set was then evaluated using the exhaustive search method with the Bhattacharyya distance measure as the selection criterion. The Bhattacharyya exhaustive search selected a different subset of features: features one and three. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms of each training set with only the selected features are shown in Figure 17.

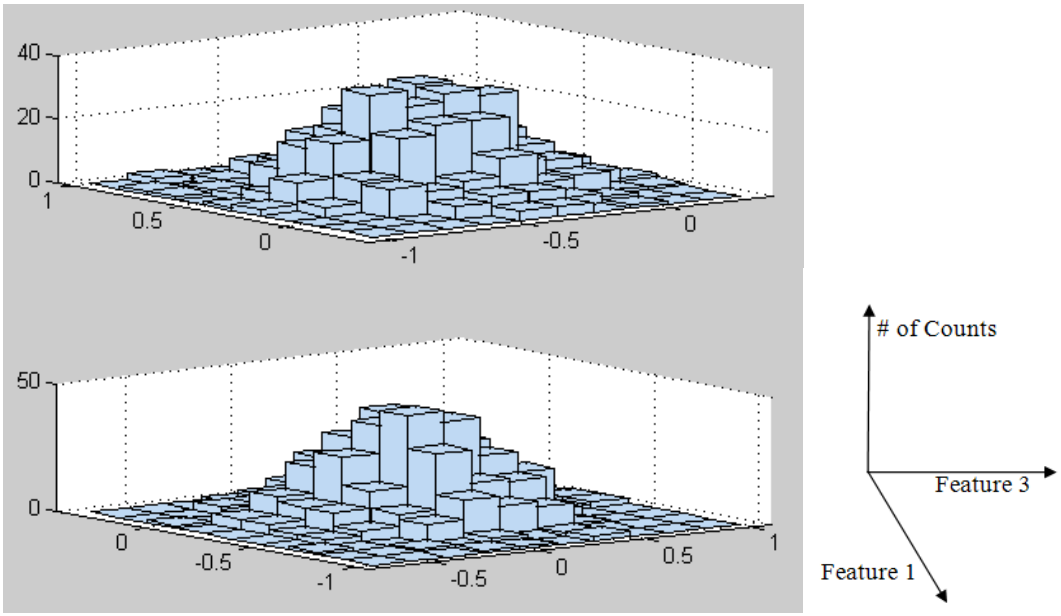

Figure 17. Two-dimensional histograms of the two classes with the selected subset of features, one and three, obtained using the exhaustive search approach with the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 17 is shown as a scatter plot in Figure 18. These two figures show the separability of the two class PDFs chosen by the Hellinger algorithm.

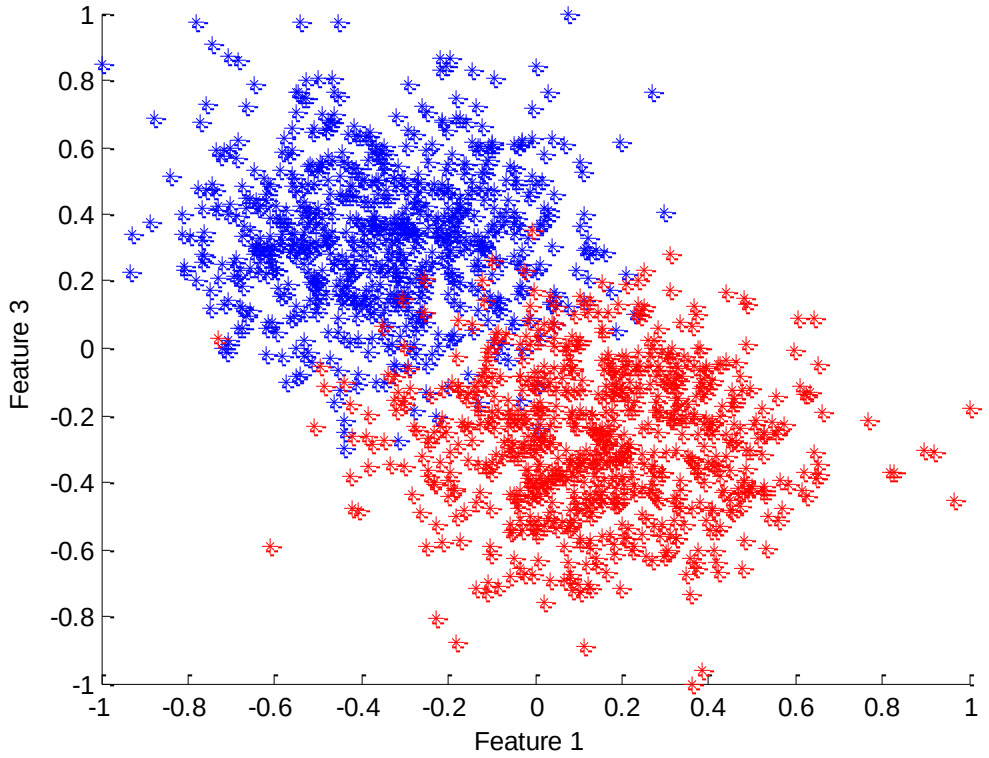

Figure 18. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 96.9% of the test data. The ROC curve for this implementation is shown in Figure 19.

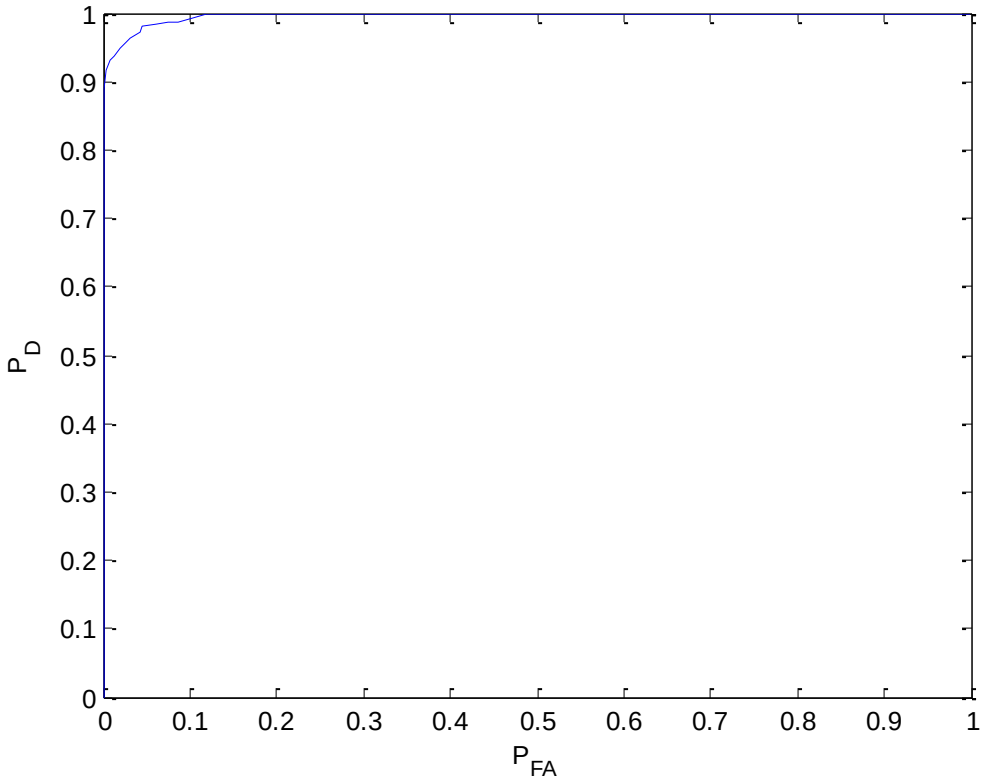

Figure 19. ROC curve obtained with the exhaustive search method and the Bhattacharyya distance for the two Gaussian distributed target classes using features one and three.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate (green line) are shown in Figure 20.

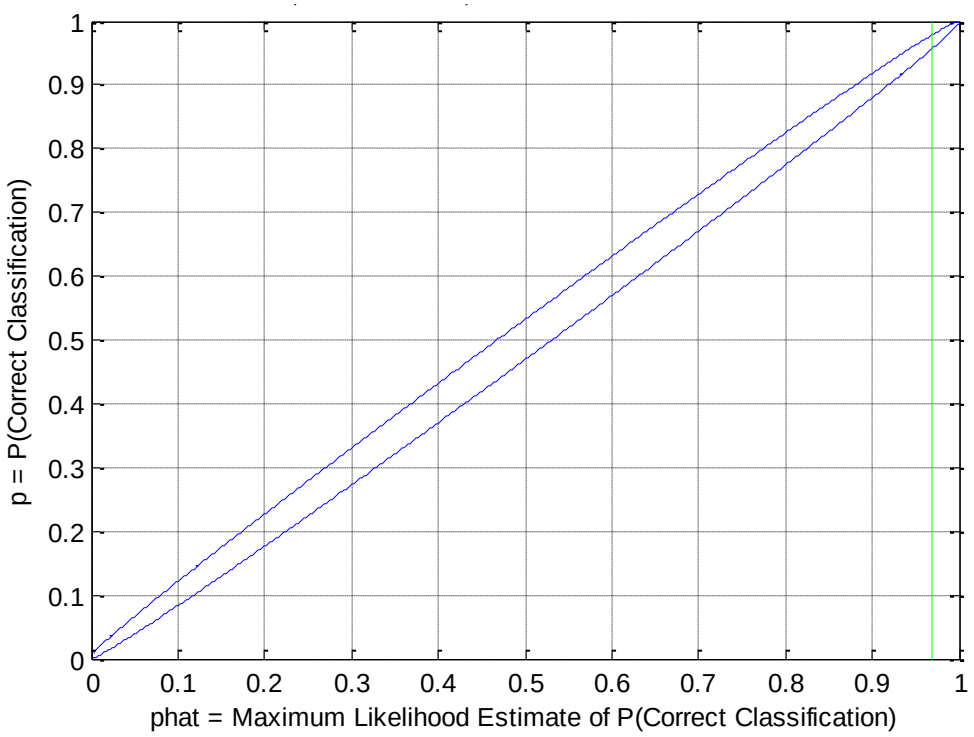

Figure 20. 95% confidence interval for the correct classification rate of 96.90% for the exhaustive search method and the Bhattacharyya distance results for a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 95.11% and an upper bound of 98.20%. The same data set was then evaluated using the exhaustive search method with the Mahalanobis distance measure as the selection criterion. The Mahalanobis algorithm selected the same subset of features as the Bhattacharyya algorithm, thereby providing the same correct classification rate.

There is a small overlap in the confidence intervals between the Hellinger and the other distance measures, showing that there is not complete confidence that the Bhattacharyya or Mahalanobis measures consistently outperform the Hellinger measure. Results showed all measures selected a feature combination resulting in a similiar correct classification rate. Since the feature distributions for both classes were Gaussian, and both the Bhattacharyya and Mahalanobis distance measures assume Gaussianity, it is expected that these algorithms will perform roughly equivalently.

2. One Gaussian Class and One Non-Gaussian Target Class

The first test of the feature selection algorithm with one Gaussian and one non-Gaussian target class involved both classes having a combined 1400 training vectors (700 per class) and 1000 test vectors (500 per class) with a total of eight available features. The best two features were then selected among the available combinations of features. The algorithm evaluated for this case was the exhaustive search method. This algorithm was implemented three times on the set of data, once for each class separability measure discussed. The distributions of each feature in feature space for each of the classes are shown in Figures 21 and 22. The distributions were calculated after each generated data set was normalized as discussed in Chapter V.

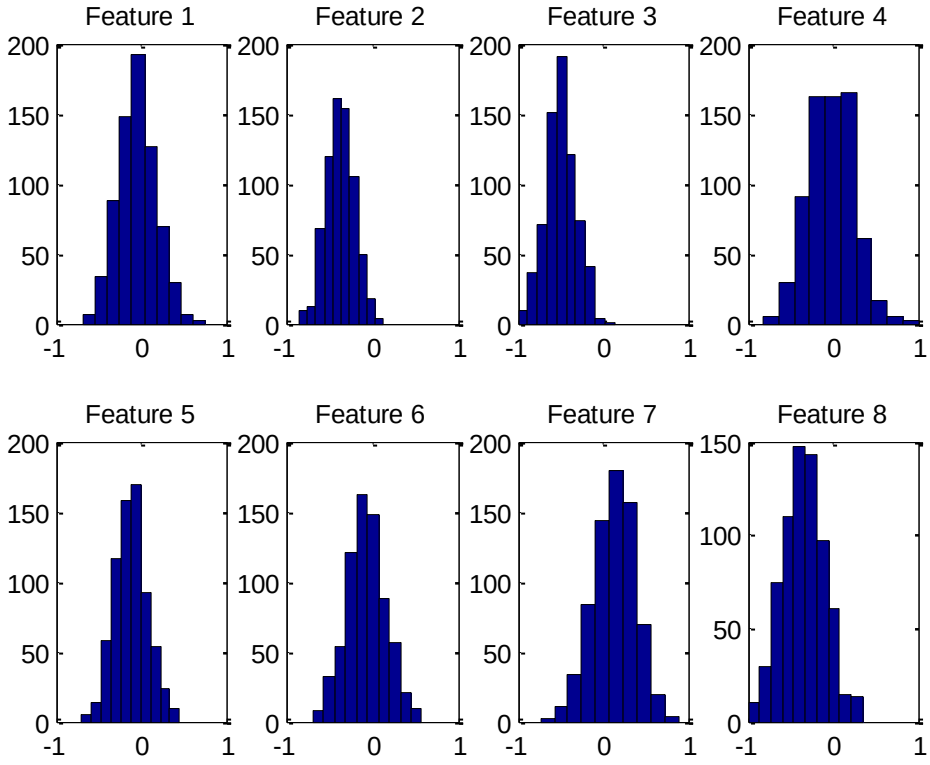

Figure 21. Feature distributions for each feature of H_0 in feature space for a Gaussian distributed target class.

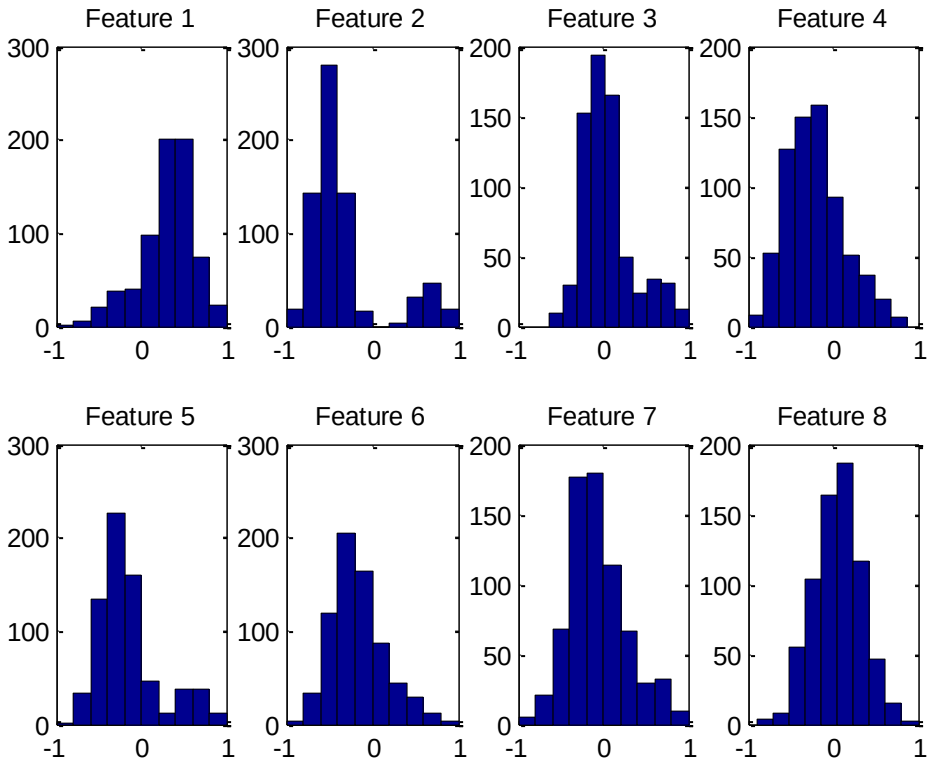

Figure 22. Feature distributions for each feature of H_1 in feature space for a non-Gaussian distributed target class.

The first class separability measure used in the exhaustive search method was the Hellinger distance measure. Since the Hellinger distance method requires the estimation of the class PDFs, the algorithm was given a grid scale factor of 1.1 and a sampling interval *dX* of 0.1. The grid scale factor was varied over a range of [1.1, 2.0] and did not affect the algorithm's results. Similarly, the sampling interval was varied between 0.0001 and 0.1 and this also did not affect the results. With the data set described above, the Hellinger Exhaustive Search algorithm selected features three and eight. This information was then used to construct the final training and test vectors as discussed in the Chapter V. The two-dimensional histograms obtained for each training set with the selected features only are shown in Figure 23.

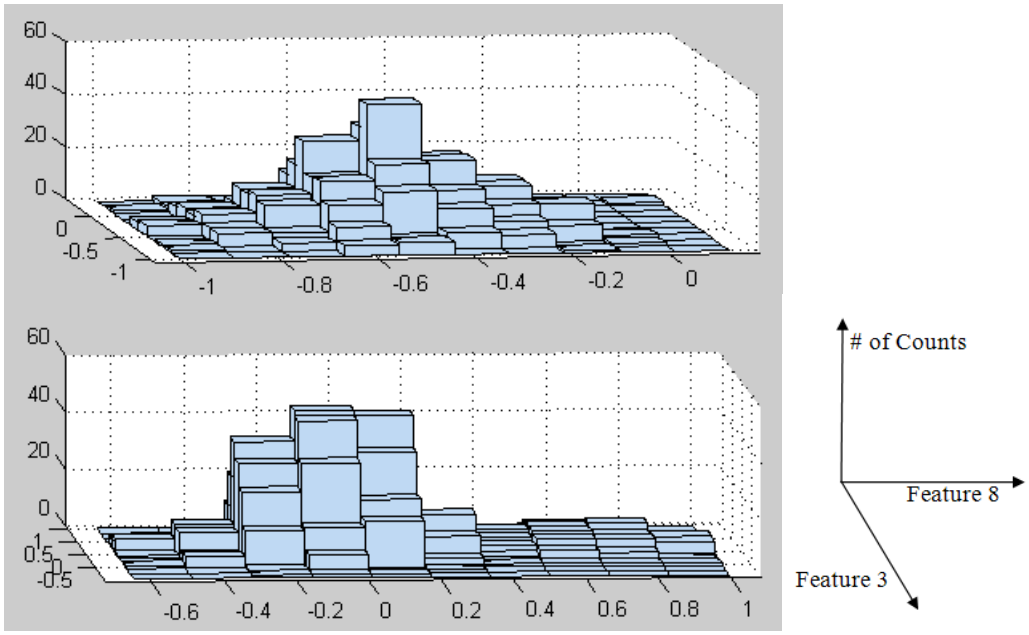

Figure 23. Two-dimensional histograms of the two classes with the selected subset of features, three and eight, obtained using the exhaustive search method and the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 23 is shown as a scatter plot in Figure 24. These two figures show the separability of the two class PDFs chosen by the Hellinger algorithm.

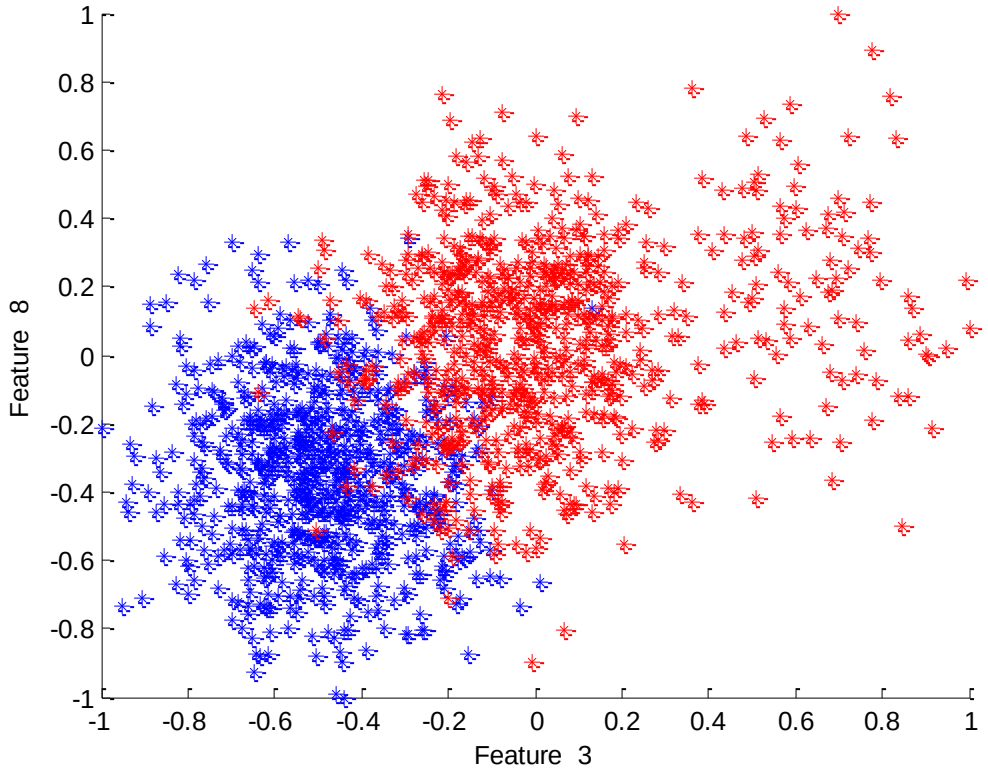

Figure 24. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 99.00% of the test data. The ROC curve for this implementation is shown in Figure 25.

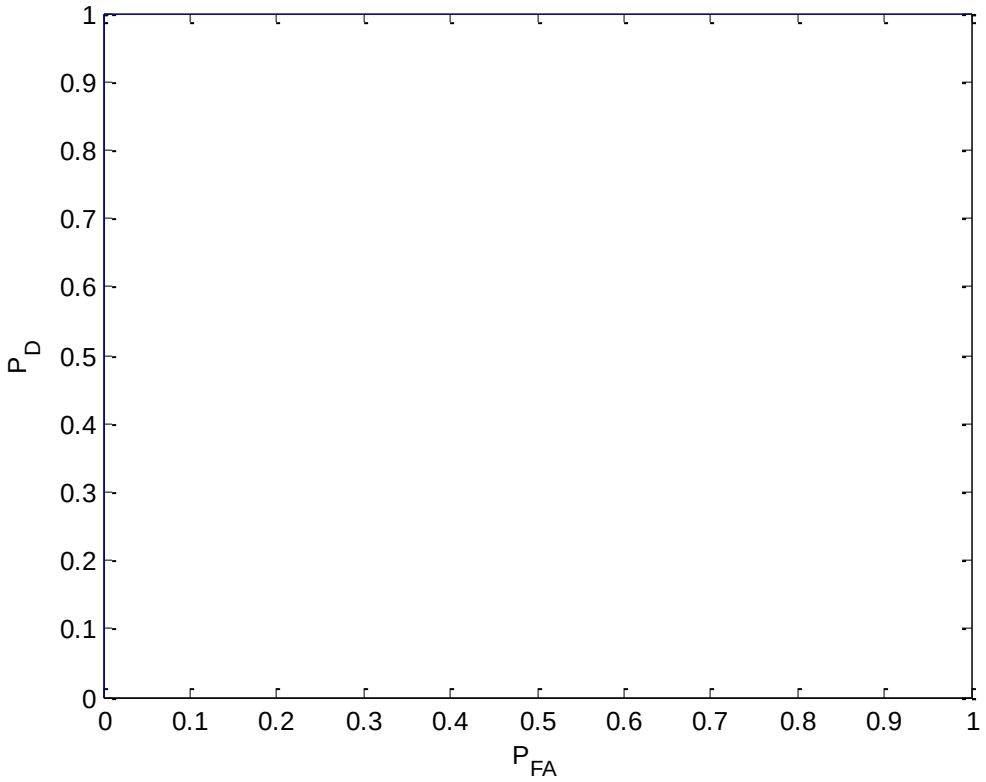

Figure 25. ROC curve obtained with the exhaustive search method and the Hellinger distance for one Gaussian and one non-Gaussian distributed target class using features three and eight.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 26.

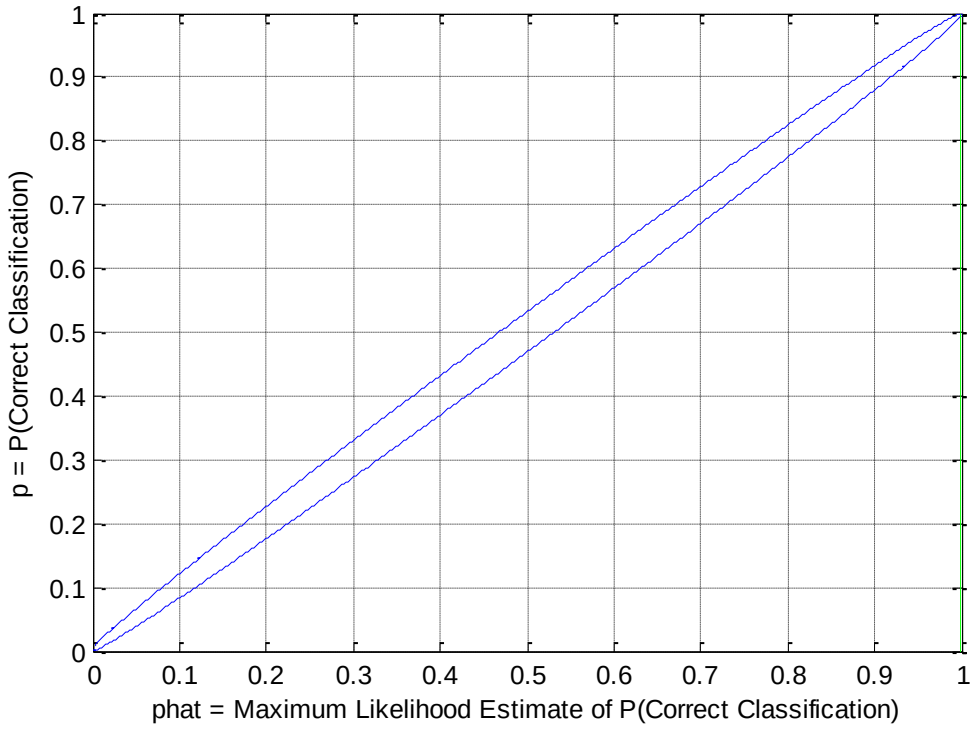

Figure 26. 95% confidence interval for the correct classification rate of 99.90% obtained for the exhaustive search and the Hellinger distance results for a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 99.42% and an upper bound of 99.98%.

The same data set was then evaluated using the exhaustive search method with the Bhattacharyya distance measure as the selection criterion. The Bhattacharyya algorithm selected a different subset of features: features one and three. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms of each training set with only the selected features are shown in Figure 27.

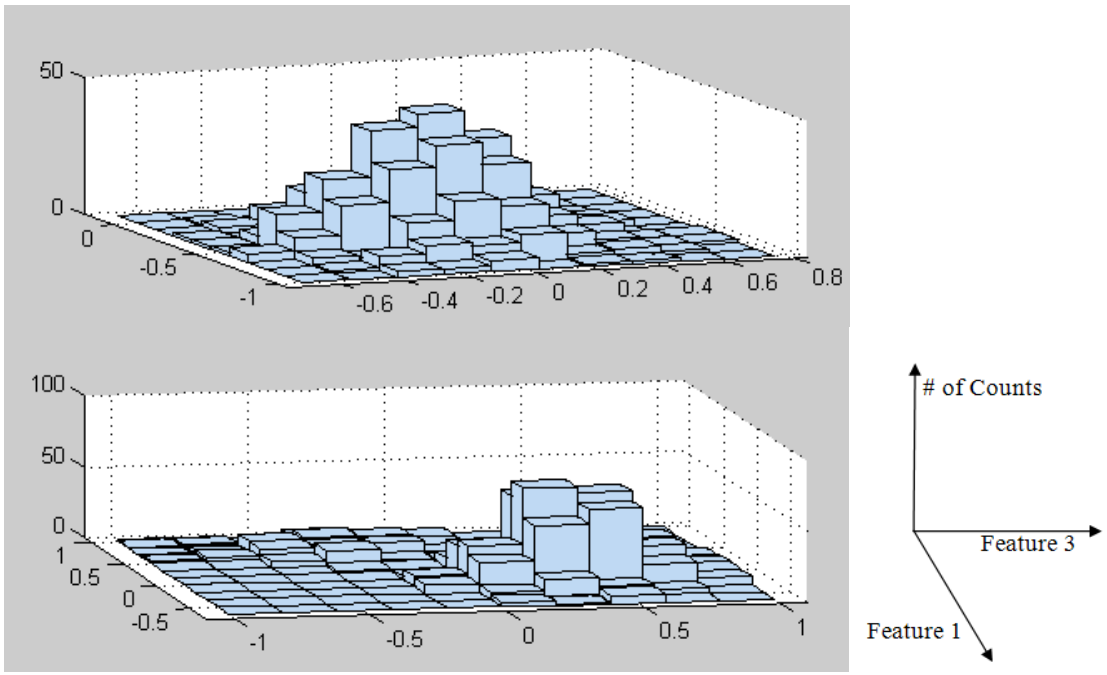

Figure 27. Two-dimensional histograms of the two classes with the selected subset of features, one and three, obtained using the exhaustive search method and the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 27 is shown as a scatter plot in Figure 28. These two figures show the separability of the two class PDFs chosen by the Bhattacharyya algorithm.

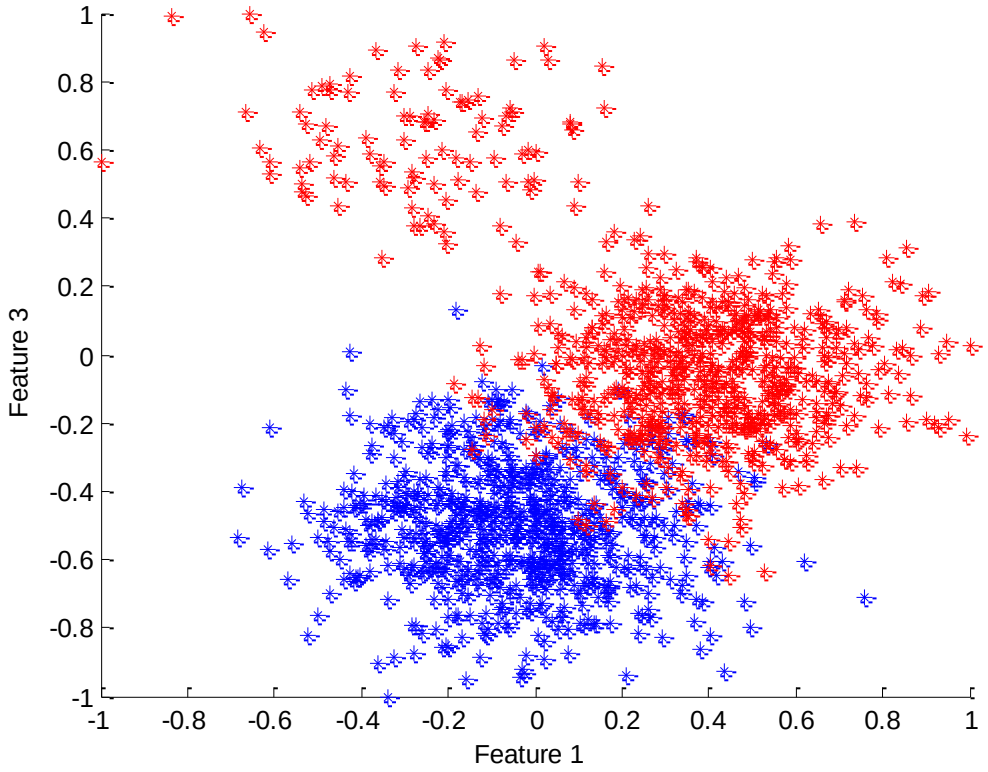

Figure 28. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 91.40% of the test data. The ROC curve for this implementation is shown in Figure 29.

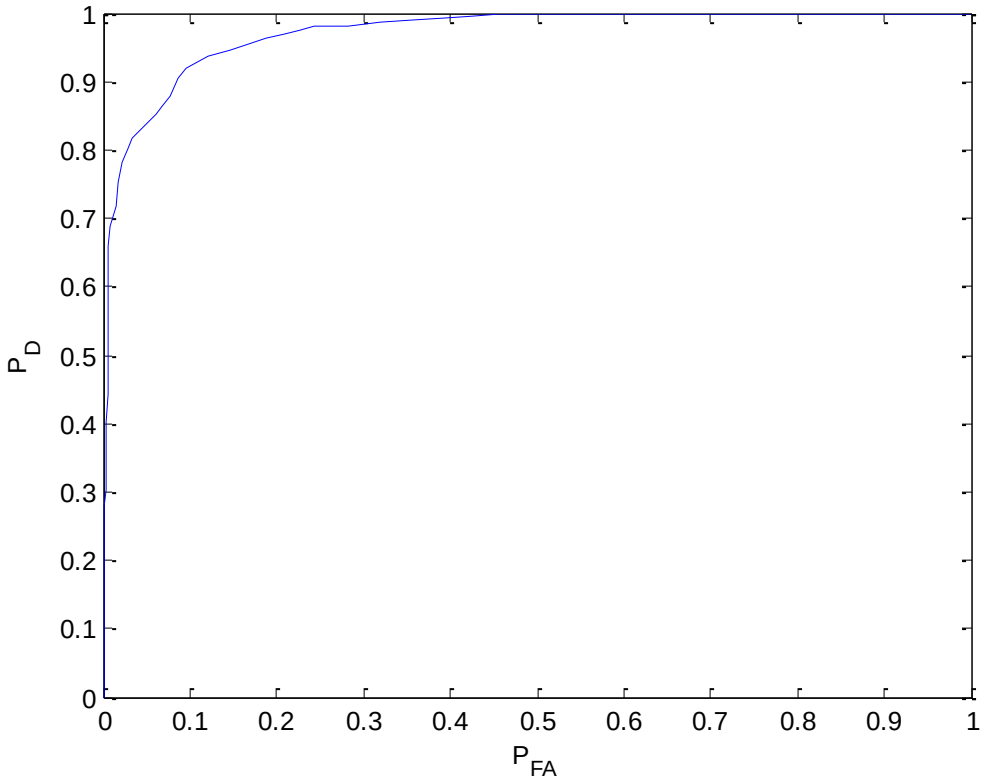

Figure 29. ROC curve obtained with the exhaustive search method and the Bhattacharyya distance for one Gaussian and one non-Gaussian distributed target class using features one and three.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data, given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 30.

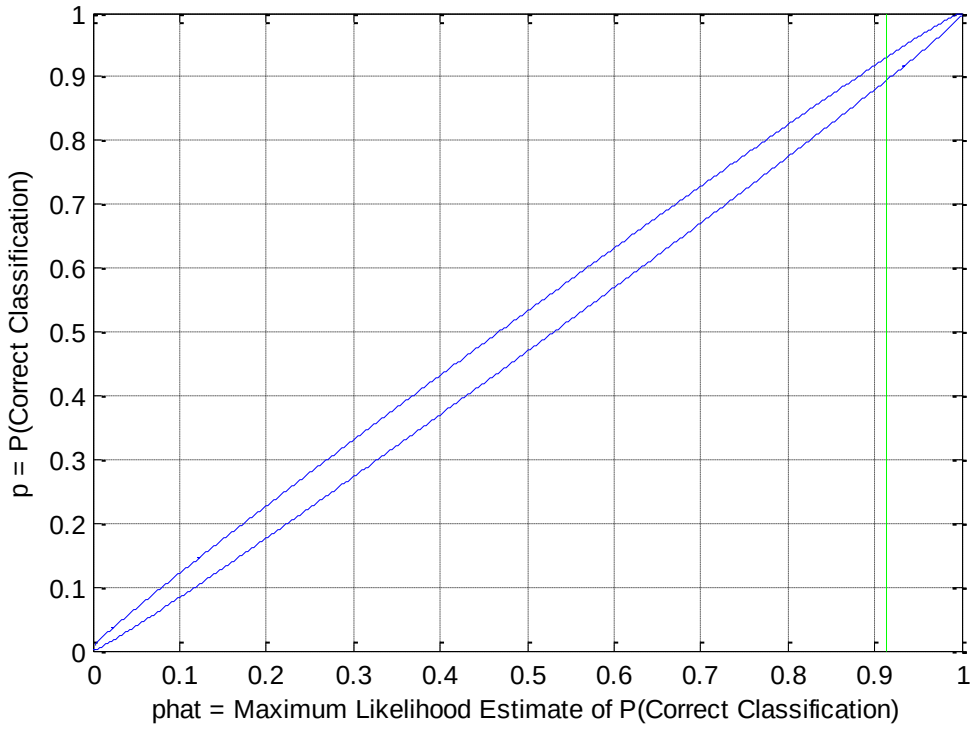

Figure 30. 95% confidence interval for the correct classification rate of 91.40% obtained for the exhaustive search method and the Bhattacharyya distance results for a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 89.46% and an upper bound of 93.01%. As expected, the Hellinger distance performed better than the Bhattacharyya distance for these data sets. The same data set was then evaluated using the exhaustive search method with the Mahalanobis distance measure as the selection criterion. The Mahalanobis algorithm selected the same subset of features as the Hellinger algorithm, thereby providing the same correct classification rate.

These data sets were then used with the SFS algorithm for each of the three separability measures. The Hellinger and Bhattacharyya distances both selected the same subset of features while the Mahalanobis SFS algorithm selected a different subset consisting of features one and three. These features are the same as those selected by the Bhattacharyya algorithms and constitute degradation in quality from the Mahalanobis

exhaustive search results. This is explained simply from the fact the SFS algorithm is not an optimal search technique and does not guarantee the optimal solution.

3. Two Non-Gaussian Distributed Target Classes

The first test of the feature selection algorithm with two non-Gaussian target classes involved both classes having a combined 940 training vectors (470 per class) and 640 test vectors (320 per class) with a total of eight available features. The best two features were then selected among the available combinations of features. The first algorithm evaluated was the exhaustive search method. This algorithm was implemented three times on the set of data, once for each class separability measure discussed. The distributions of each feature in feature space for each of the classes are shown in Figures 31 and 32. The distributions were calculated after each generated data set was normalized, as discussed in Chapter V.

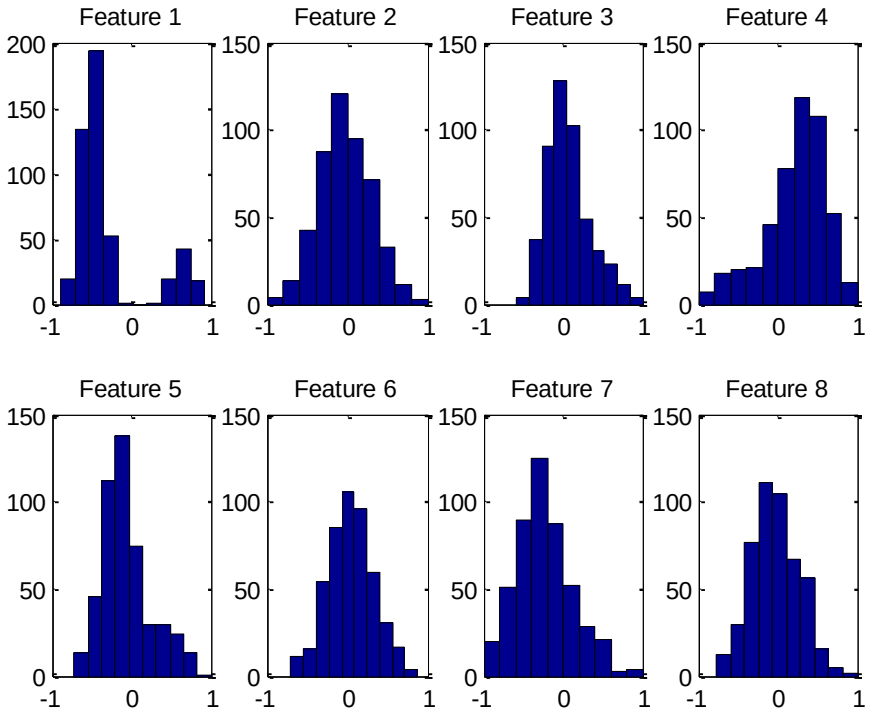

Figure 31. Feature distributions for each feature of H_0 in feature space for a non-Gaussian distributed target class.

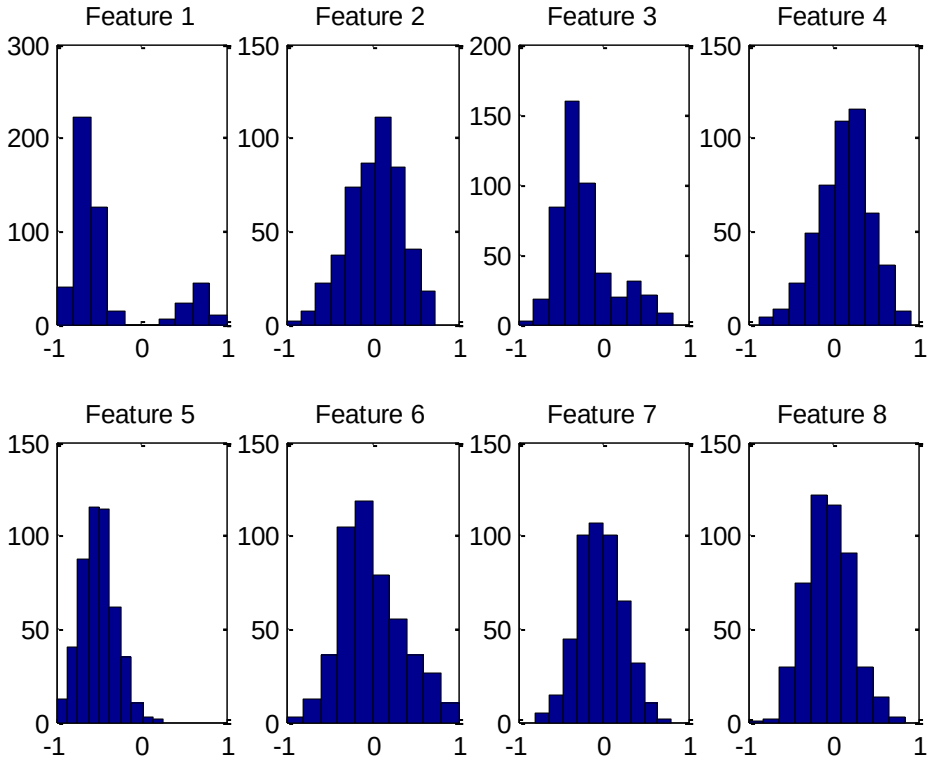

Figure 32. Feature distributions for each feature of H_1 in feature space for a non-Gaussian distributed target class.

The first class separability measure used in the exhaustive search method was the Hellinger distance measure. Since the Hellinger distance method requires the estimation of the class PDFs, the algorithm was given a Grid Scale Factor of 1.1 and a sampling interval *dX* of 0.1. The grid scale factor was varied over a range of [1.1, 2.0] and did not affect the algorithm's results. Similarly, the sampling interval was varied between 0.0001 and 0.1 and this also did not affect the results. With the data set described above, the Hellinger Exhaustive Search algorithm selected features three and five. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms for each training set, with only the selected features, are shown in Figure 33.

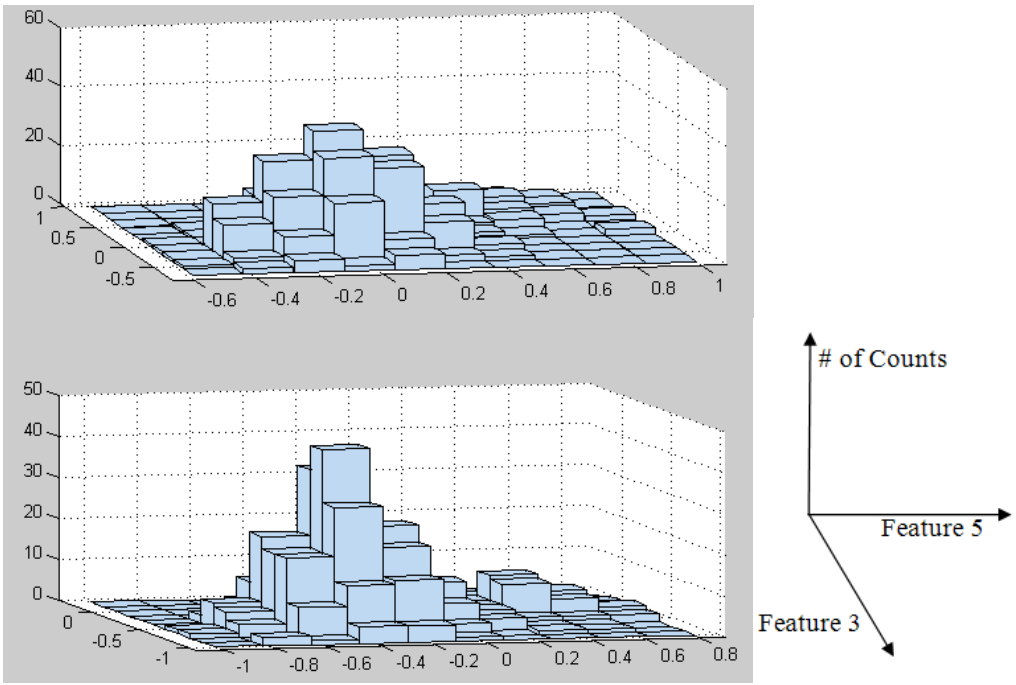

Figure 33. Two-dimensional histograms of the two classes with the selected subset of features, three and five, obtained with the exhaustive search method and the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 33 is shown as a scatter plot in Figure 34. These two figures show the separability of the two class PDFs chosen by the Hellinger algorithm.

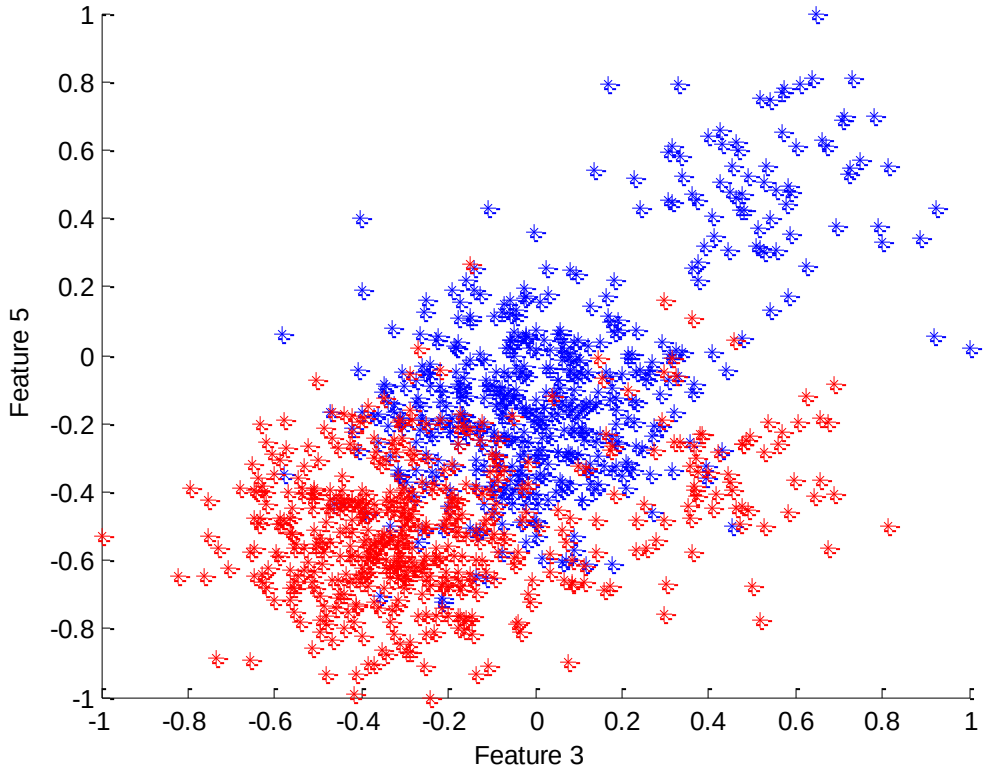

Figure 34. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 93.6% of the test data. The ROC curve for this implementation is shown in Figure 35.

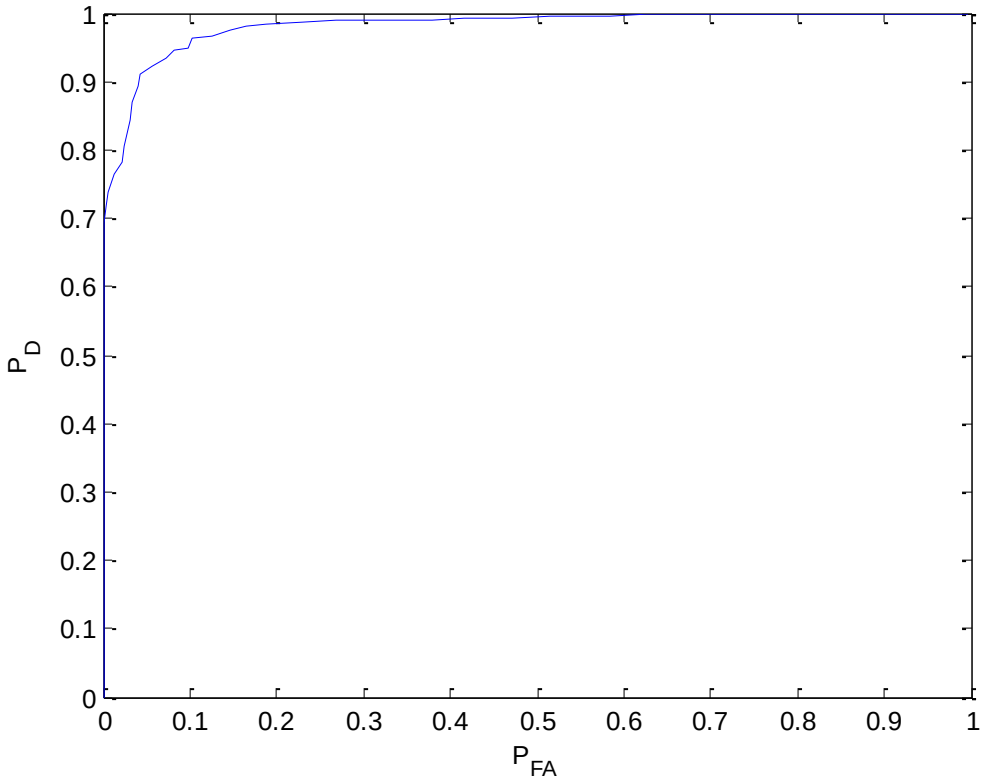

Figure 35. ROC curve obtained with the exhaustive search method and the Hellinger distance for the two non-Gaussian distributed target classes using features three and five.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 36.

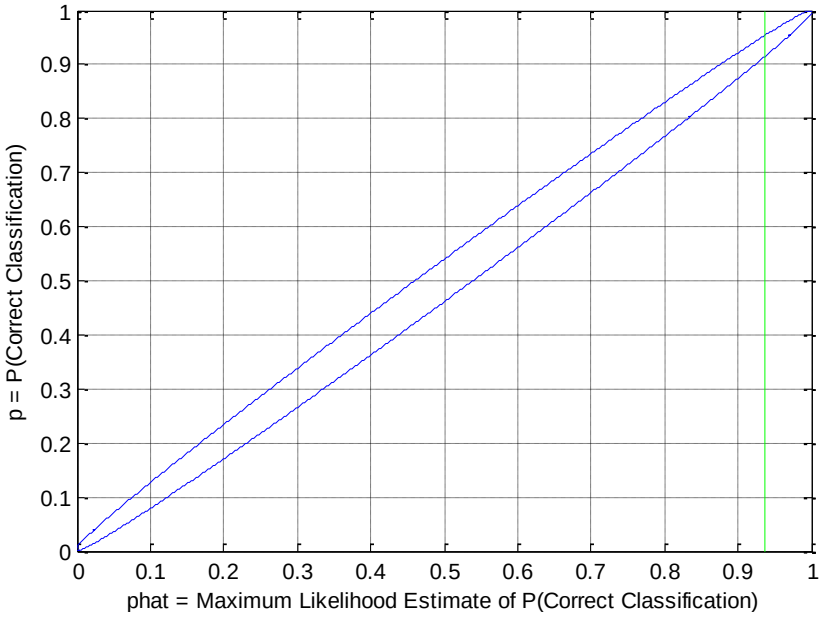

Figure 36. 95% confidence interval for the correct classification rate of 93.6% obtained for the exhaustive search method using the Hellinger distance results for a test set size of 640 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 91.37% and an upper bound of 95.27%. The same data set was then evaluated using the exhaustive search method with the Bhattacharyya distance measure as the selection criterion. The Bhattacharyya algorithm selected a different subset of features: features five and seven. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms of each training set, with only the selected features, are shown in Figure 37.

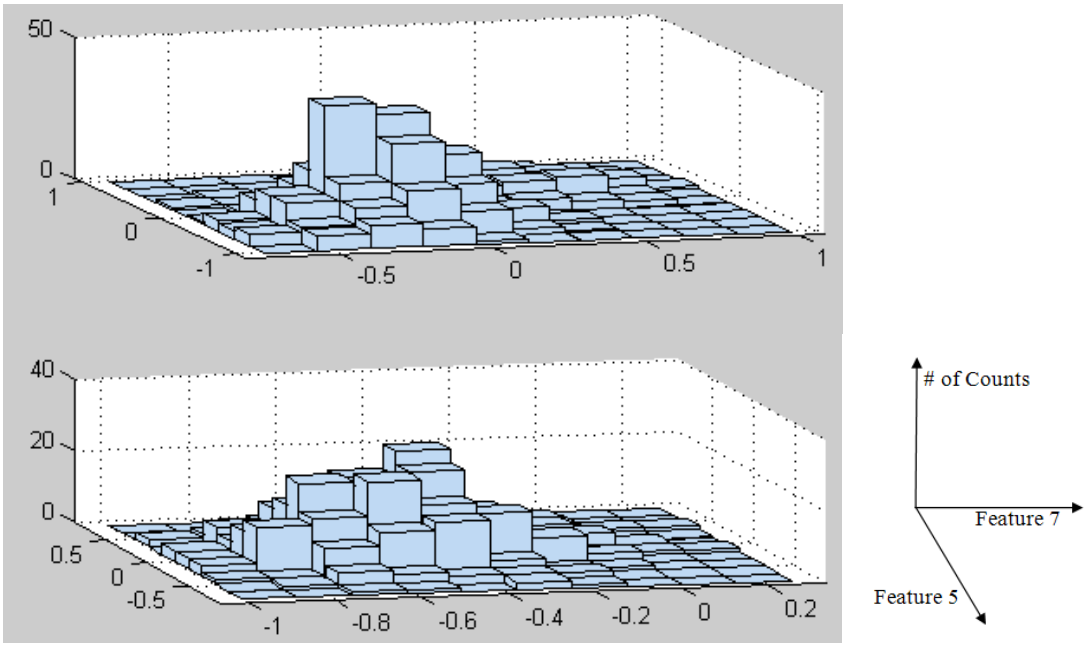

Figure 37. Two-dimensional histograms of the two classes with the selected subset of features, five and seven, obtained using the exhaustive search approach and the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 37 is shown as a scatter plot in Figure 38. These two figures show the separability of the two class PDFs chosen by the Bhattacharyya algorithm.

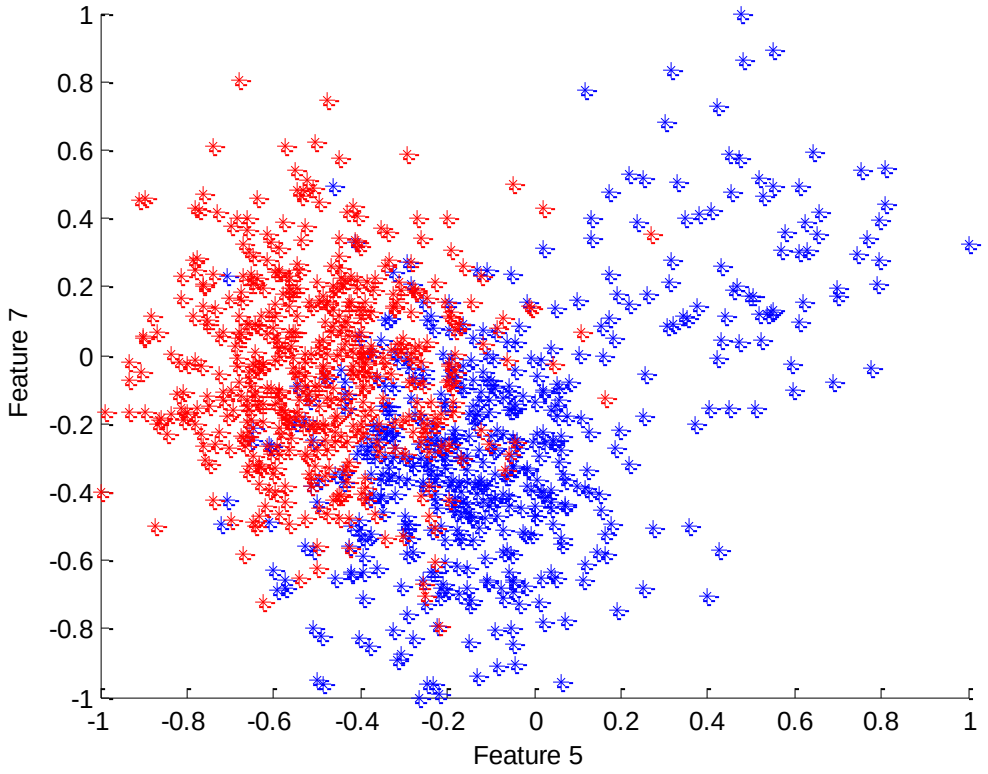

Figure 38. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 88.6% of the test data. The ROC curve for this implementation is shown in Figure 39.

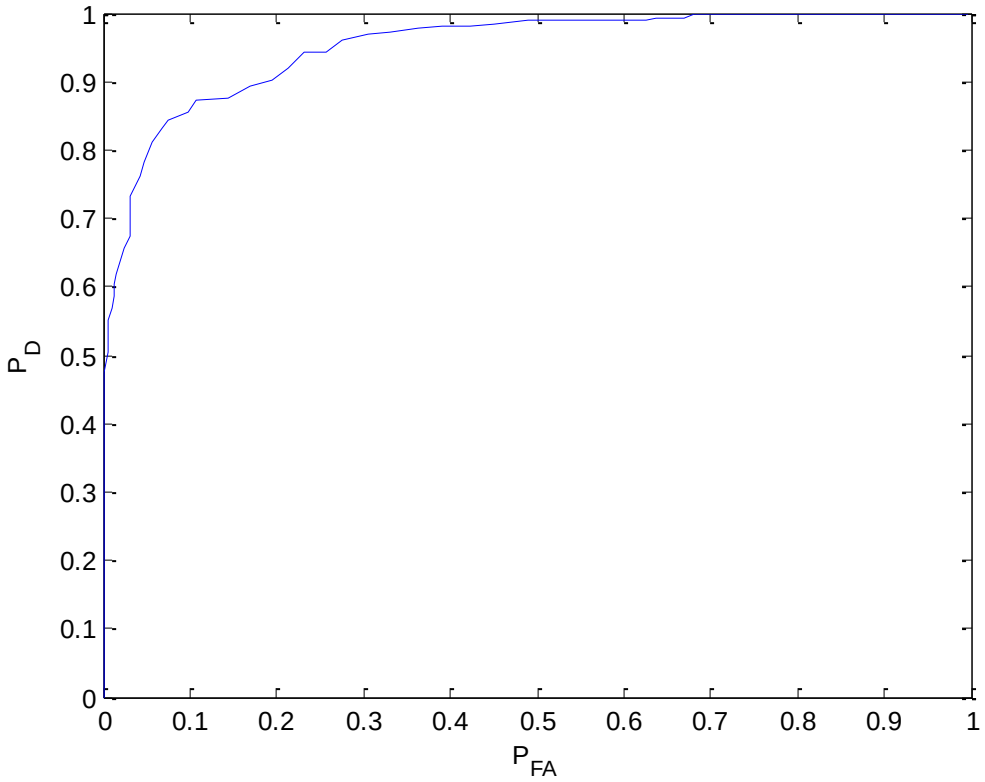

Figure 39. ROC curve obtained with the exhaustive search method and the Bhattacharyya distance for the two non-Gaussian distributed target classes using features five and seven.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 40.

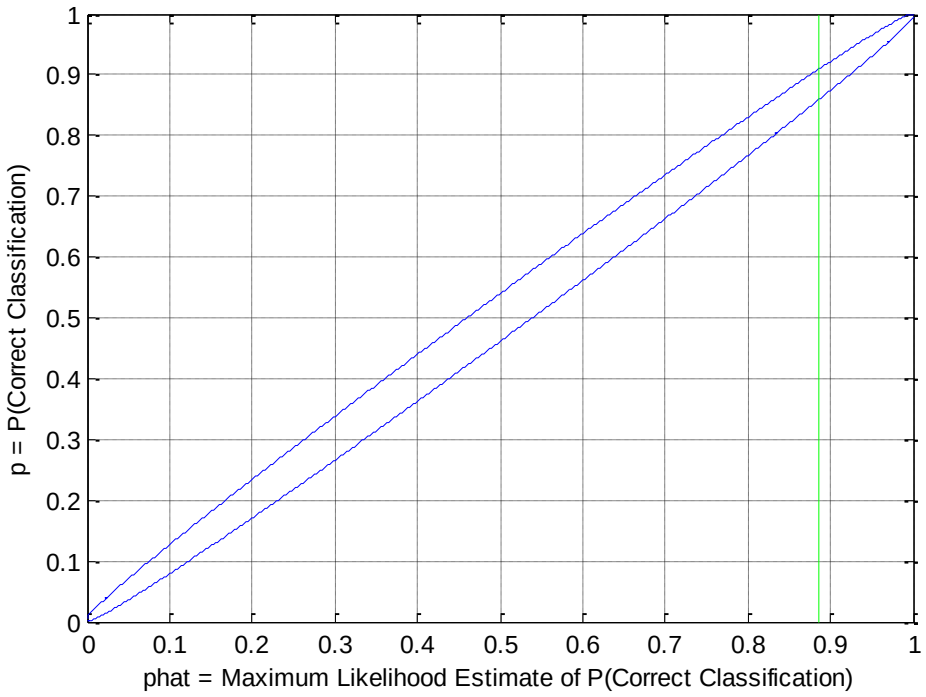

Figure 40. 95% confidence interval for the correct classification rate of 88.6% obtained for the exhaustive search method using the Bhattacharyya distance results for a test set size of 640.

The evaluation of the confidence interval with these parameters yielded a lower bound of 85.84% and an upper bound of 90.87%. The same data set was then evaluated using the exhaustive search method with the Mahalanobis distance measure as the selection criterion. The Mahalanobis algorithm selected a different subset of features: features three and four. With this subset of features, the Bayes classifier provided a maximum correct classification rate of 58.9% of the test data. The ROC curve for this implementation is shown in Figure 41.

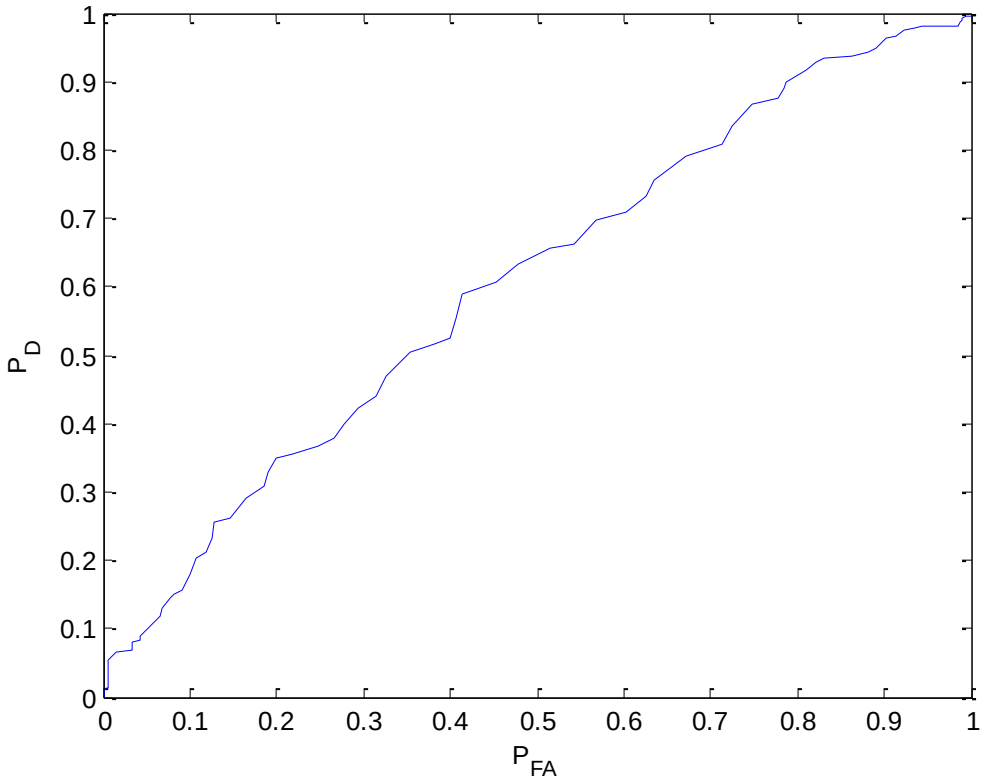

Figure 41. ROC curve obtained with the exhaustive search method and the Mahalanobis distance for the two non-Gaussian distributed target classes using features three and four.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 42.

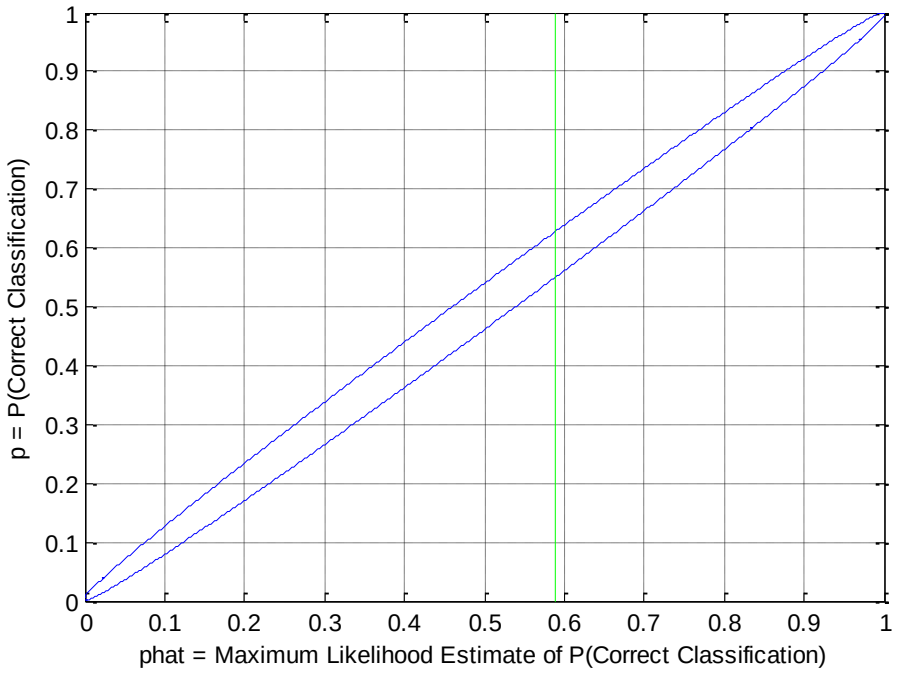

Figure 42. 95% confidence interval for the correct classification rate of 58.9% for the exhaustive search using the Mahalanobis distance results with a test set size of 640 vectors.

The results of the Mahalanobis exhaustive search are not considered valid because many of the feature subset combinations provided singular or close to singular matrices during the distance measure evaluation. The combination of two separate Gaussians caused the covariance matrix to be non-diagonal and ill-conditioned for the distance evaluation. This is a result of the manner in which the data were generated, as discussed in Chapter V. After all three separability measures were used in the algorithm, it was evident that the Hellinger distance was most capable of selecting the best non-Gaussian feature combination. The confidence intervals obtained for the Bhattacharyya and Mahalanobis algorithms do not overlap with that obtained for the Hellinger, providing more confidence in these positive results.

This data set was then used in the SFS algorithm with all three class separability measures. The first distance measure evaluated was the Hellinger. The SFS algorithm utilizing the Hellinger distance as the selection criterion yielded the same results as the exhaustive search with the Hellinger distance: features three and five. The same feature

combination generated the same correct classification rate of 93.6%. The SFS algorithm for the Bhattacharyya also selected the same feature subset as the exhaustive search: features five and seven. The only distance measure that saw a change in the selected feature subset between the exhaustive search and the SFS was the Mahalanobis. The Mahalanobis selected features five and seven, the same subset as the Bhattacharyya, yielding a correct classification rate of 88.6%.

In an effort to better evaluate the algorithms and to test if a sufficient number of points were used, another set of non-Gaussian distributed target classes was generated. This trial contained a combined 1400 training vectors and 1000 test vectors with a total of eight available features. The best two features were then selected among the available combinations of features. The exhaustive search method was once again implemented with all three class separability measures. The distributions of each feature in feature space for each of the classes are shown in Figures 43 and 44. The distributions were calculated after each generated data set was normalized as discussed in Chapter V.

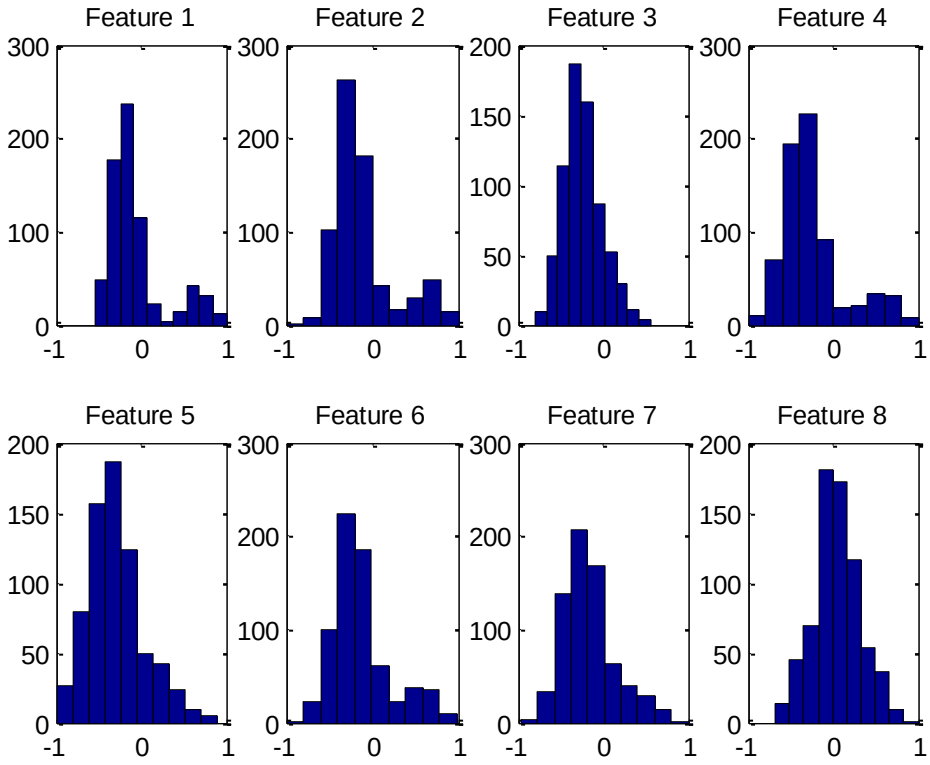

Figure 43. Feature distributions for each feature of H_0 in feature space for a non-Gaussian distributed target class.

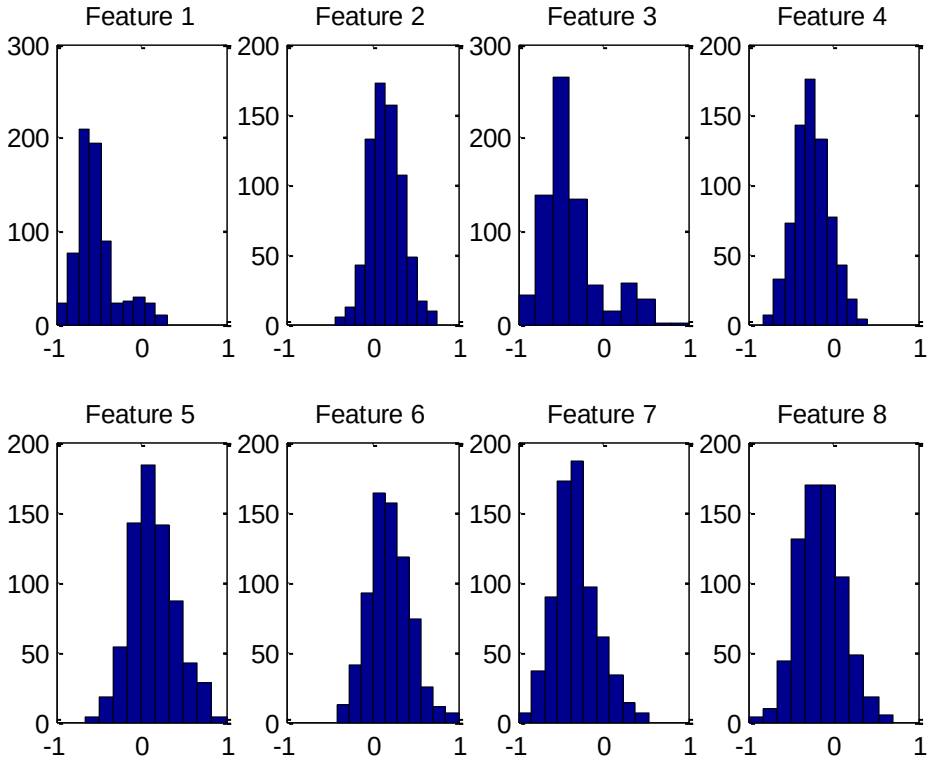

Figure 44. Feature distributions for each feature of H_1 in feature space for a non-Gaussian distributed target class.

The first class separability measure used in the exhaustive search method was the Hellinger distance measure. Since the Hellinger distance method requires the estimation of the class PDFs, the algorithm was given a Grid Scale Factor of 1.1 and a sampling interval *dX* of 0.1. The grid scale factor was varied over a range of [1.1, 2.0] and did not affect the algorithm's results. Similarly, the sampling interval was varied between 0.0001 and 0.1 and this also did not affect the results. With the data set described above, the Hellinger Exhaustive Search algorithm selected features one and five. This information was then used to construct the final training and test vectors as discussed in the Experimental Setup chapter. The two-dimensional histograms obtained for each training set with only the selected features are shown in Figure 45.

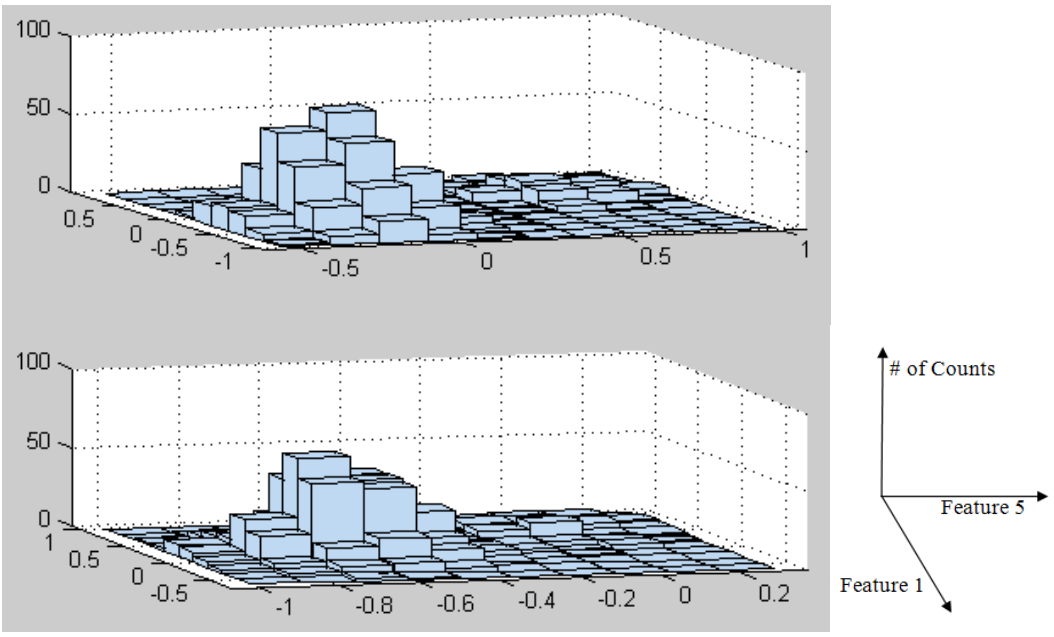

Figure 45. Two-dimensional histograms of the two classes with the selected subset of features, one and five, obtained with the exhaustive search method and the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 45 is shown as a scatter plot in Figure 46. These two figures show the separability of the two class PDFs chosen by the Hellinger exhaustive search algorithm.

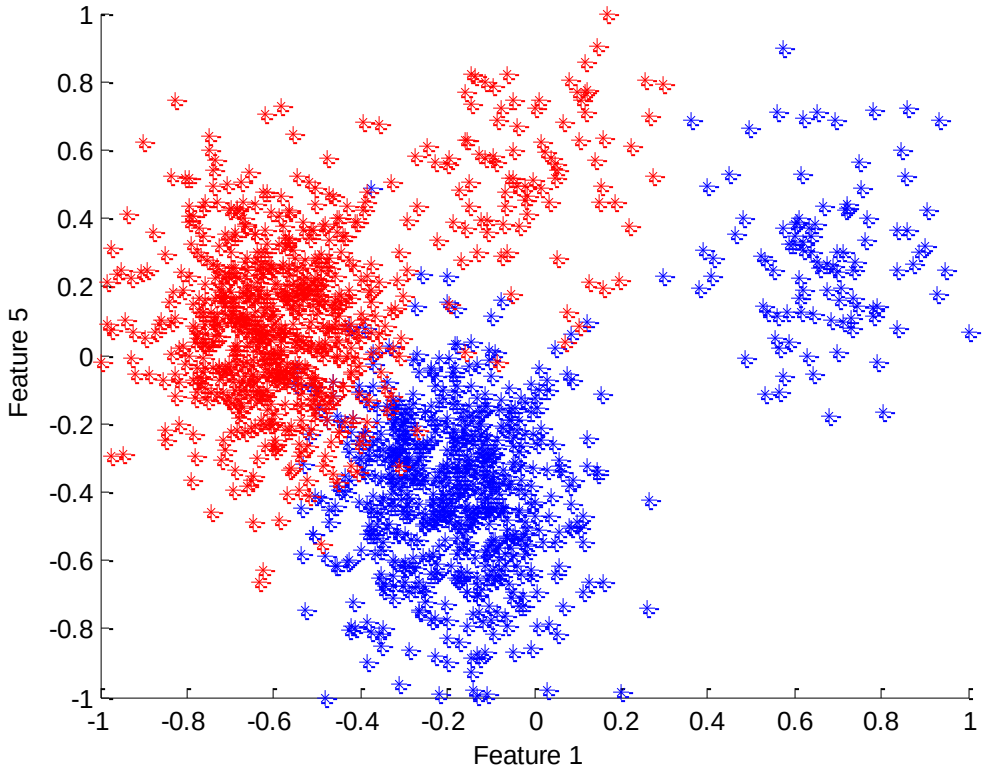

Figure 46. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 95.1% of the test data. The ROC curve for this implementation is shown in Figure 47.

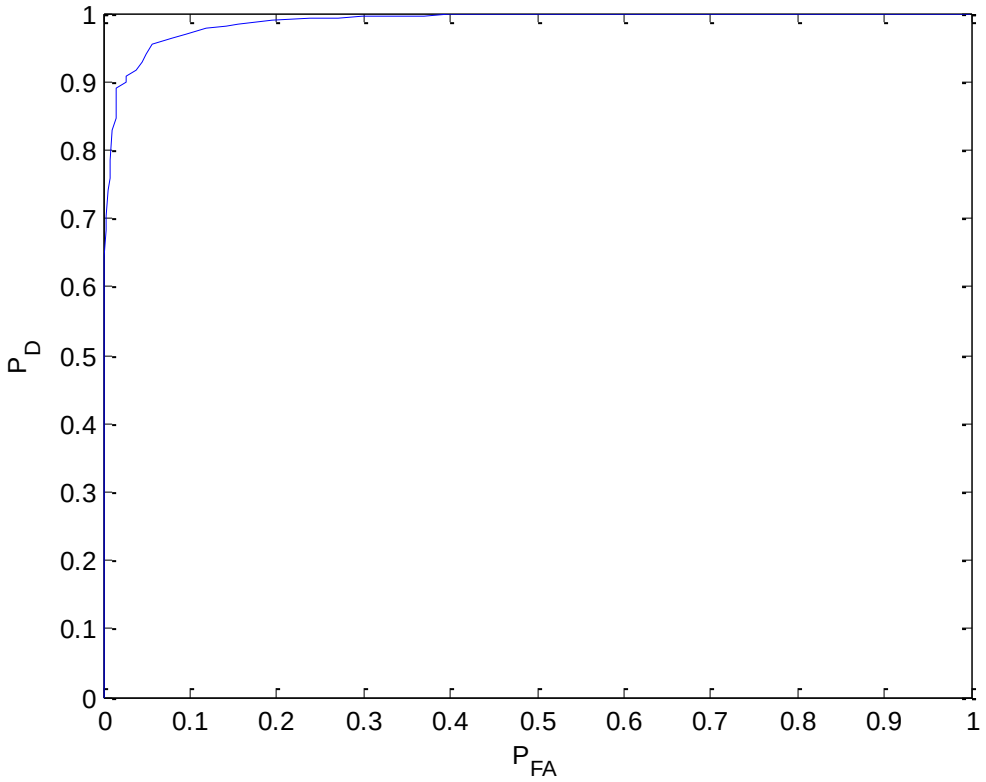

Figure 47. ROC curve obtained using the exhaustive search with the Hellinger distance for the two non-Gaussian distributed target classes using features one and five.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 48.

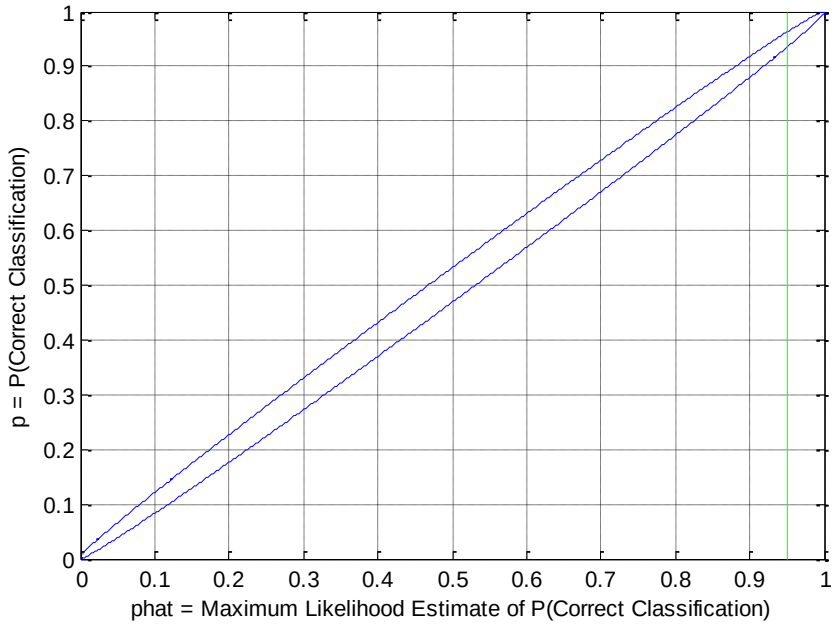

Figure 48. 95% confidence interval for the correct classification rate of 95.1% obtained for the exhaustive search method using the Hellinger distance results with a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 93.55% and an upper bound of 96.29%. The same data set was then evaluated using the exhaustive search method with the Bhattacharyya distance measure as the selection criterion. The Bhattacharyya algorithm selected a different subset of features: features one and six. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms for each training set with only the selected features are shown in Figure 49.

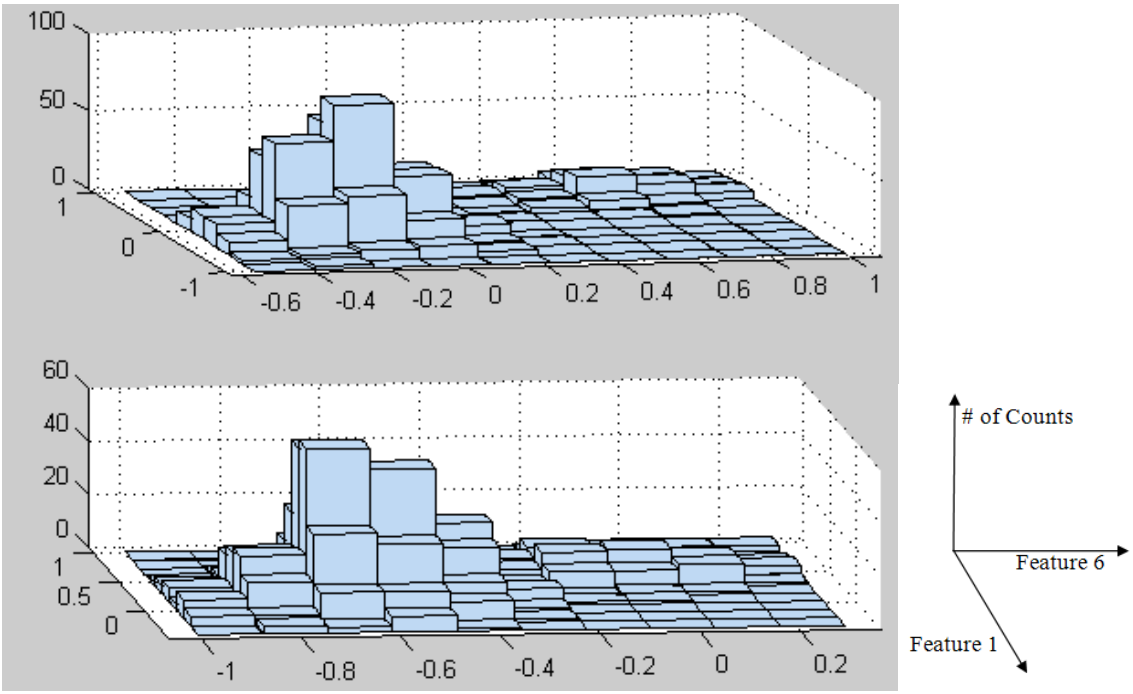

Figure 49. Two-dimensional histograms of the two classes with the selected subset of features, one and six, obtained with the exhaustive search method and the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 49 is shown as a scatter plot in Figure 50. These two figures show the separability of the two class PDFs chosen by the Bhattacharyya exhaustive search algorithm.

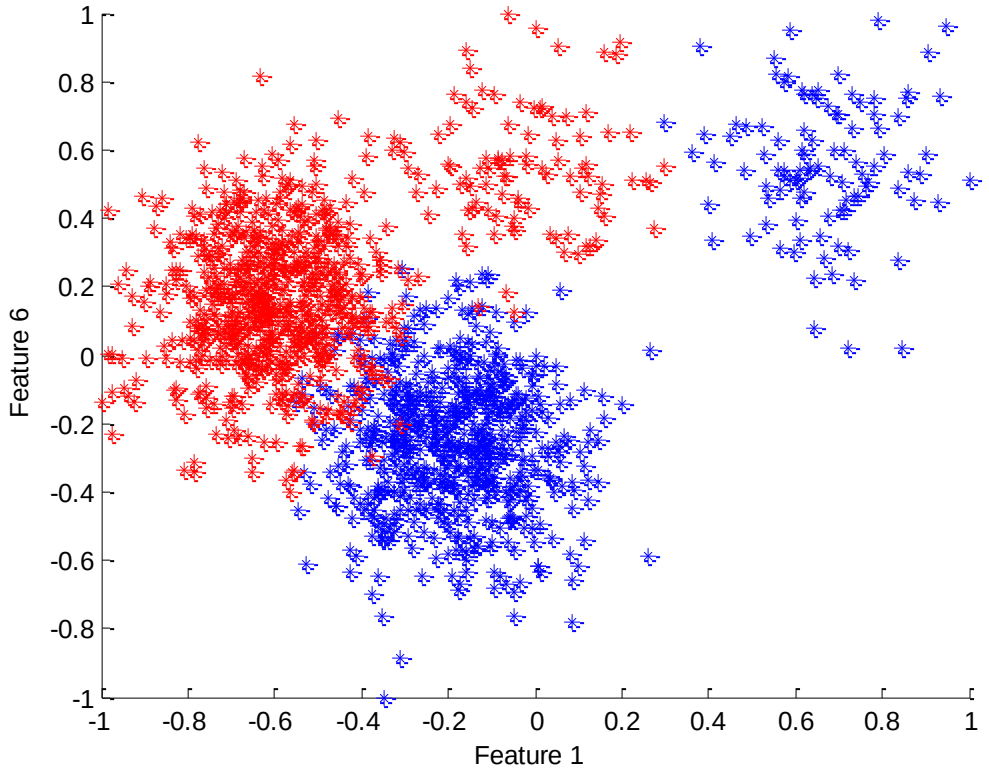

Figure 50. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 91.7% of the test data. The ROC curve for this implementation is shown in Figure 51.

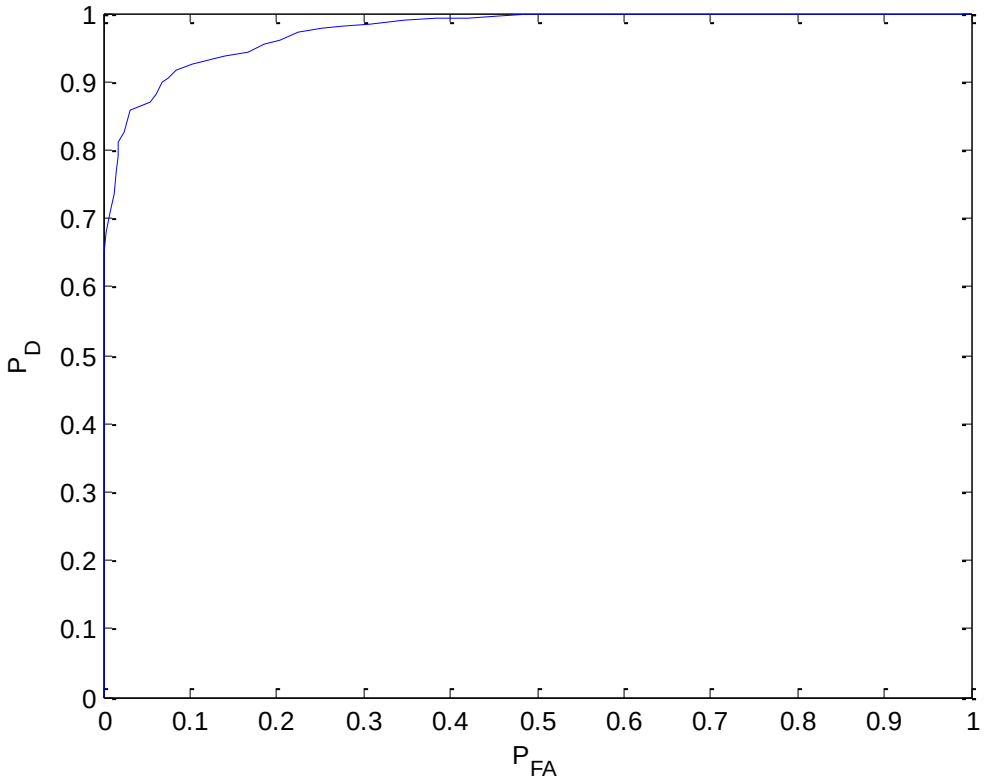

Figure 51. ROC curve obtained with the exhaustive search method and the Bhattacharyya distance for the two non-Gaussian distributed target classes using features one and six.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 52.

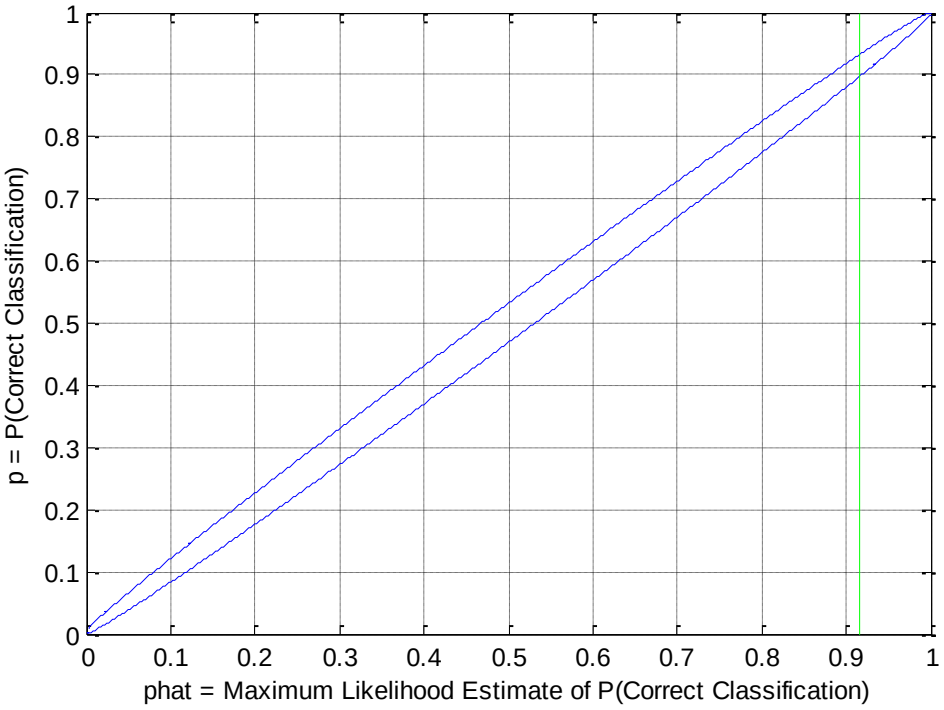

Figure 52. The 95% confidence interval for the correct classification rate of 91.7% with a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 89.78% and an upper bound of 93.28%. The same data set was then evaluated using the exhaustive search method with the Mahalanobis distance measure as the selection criterion. The Mahalanobis algorithm selected a different subset of features: features four and six. With this subset of features, the Bayes classifier provided a maximum correct classification rate of 50.0% of the test data. Just as before, these results are inconclusive due to the near singularity or near singularity of the matrices used during the distance measure evaluation.

These results using a larger sample size provided another validation that the Hellinger algorithm chose the best feature subset among the possible feature combinations. The confidence intervals do not overlap, and the increase in the correct classification rate is considered significant.

This data set was then used in the SFS algorithm with all three class separability measures. The first distance measure evaluated was the Hellinger. The SFS algorithm

utilizing the Hellinger distance as the selection criterion yielded the same results as the exhaustive search with the Hellinger distance: features one and five. The same feature combination generated the same correct classification rate of 95.1%. The SFS algorithm for the Bhattacharyya also selected the same feature subset as the exhaustive search: features one and six. The only distance measure that saw a change in the selected feature subset between the exhaustive search and the SFS was the Mahalanobis. The Mahalanobis selected features one and six, the same subset as the Bhattacharyya, yielding a correct classification rate of 91.7%.

In an effort to further validate these results, the simulated data distribution for each feature remained the same (mean vectors and covariance matrices) while the selected data points were regenerated. The mean vectors and covariance matrices of each Gaussian in the summation remained constant for each of the following tests. This essentially gives 1000 new test points as well as 1400 new training points for the same two classes. These tests were run five times to verify the results were consistent. The results are shown below.

Table 1. The resultant feature subsets and corresponding correct classification rates (P_{CC}) for five simulations of generated data are shown in this table. Each row represents a different simulation.

Mahalanobis		**Bhattacharyya**		**Hellinger**	
Features Selected	**Maximum Pcc**	**Features Selected**	**Maximum Pcc**	**Features Selected**	**Maximum Pcc**
[2 7]	96.5%	[1 5]	81.0%	[1 2]	99.5%
[3 4]	72.9%	[1 5]	81.4%	[1 7]	98.9%
[2 8]	73.1%	[1 5]	83.0%	[1 7]	99.2%
[2 3]	74.2%	[1 5]	85.6%	[1 5]	85.6%
[1 7]	98.9%	[1 5]	79.3%	[1 7]	98.9%

The exhaustive search was the feature selection algorithm used in Table 1. Again, the Mahalanobis distance algorithm results are inconclusive because of the near

singularity in the data matrices during evaluation. The Hellinger distance did not always pick the same feature subsets but consistently outperformed the other algorithms while never doing worse.

The last algorithm to be evaluated was the Branch and Bound with each of the three separability measures. The next set of tests was run on two simulated non-Gaussian target classes with five features instead of eight due to the need for the B&B to generate multiple three- and four-dimensional PDFs. The results are shown below for the Branch and Bound, and the data were also run through the Exhaustive Search to determine if the selected features are the globally optimal subset given the respective separability measure. This trial contained a combined 1400 training vectors (700 per class) and 1000 test vectors (500 per class). The best two features were selected among the available combinations of features. The distributions of each feature in feature space for each of the classes are shown in Figures 53 and 54. The distributions were calculated after each generated data set was normalized, as discussed in Chapter V.

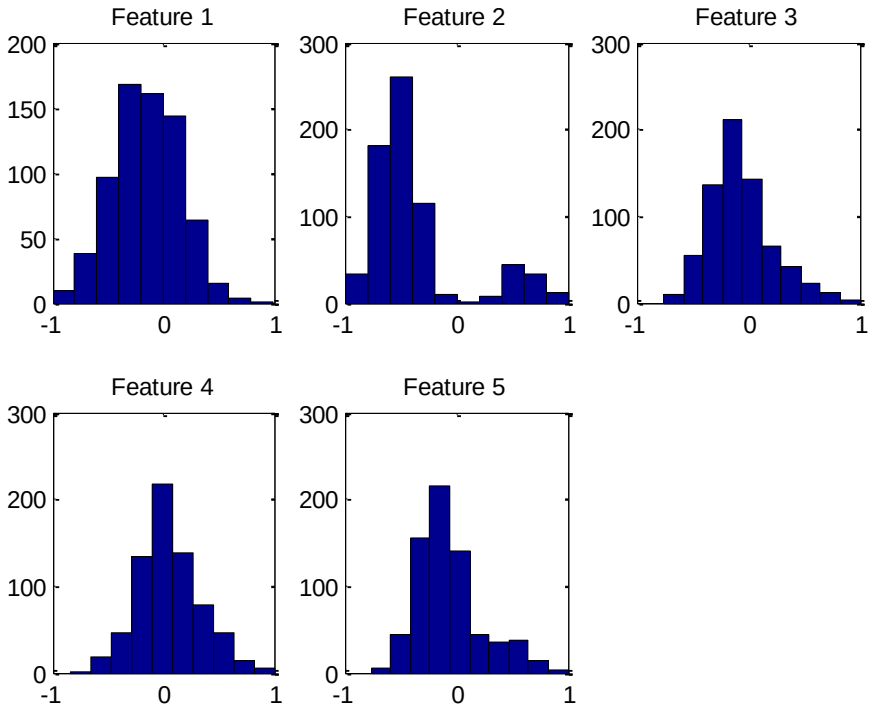

Figure 53. Feature distributions for each feature of H_0 in feature space for a non-Gaussian distributed target class.

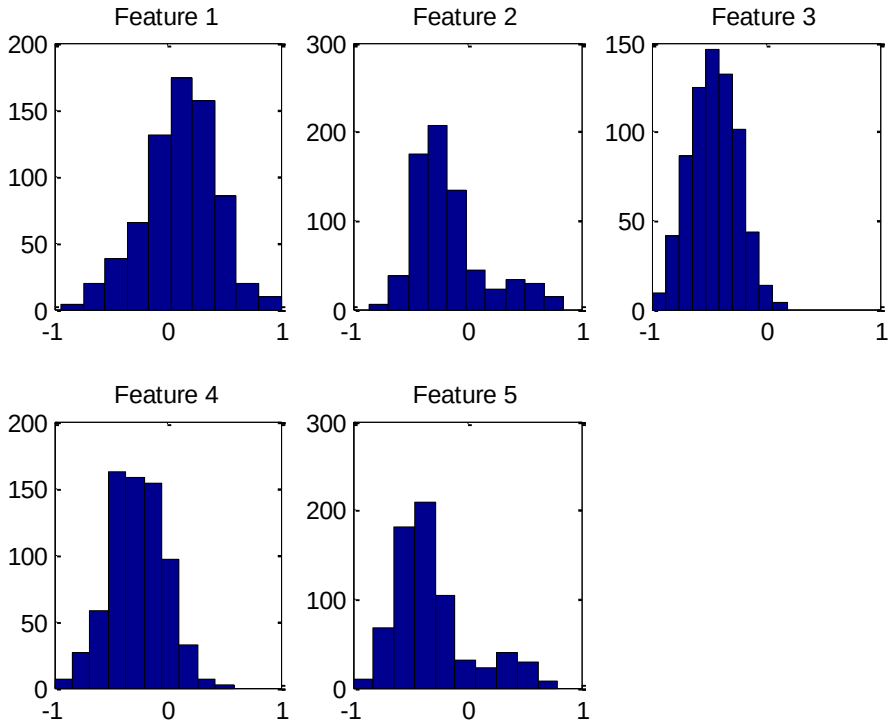

Figure 54. Feature distributions for each feature of H_1 in feature space for a non-Gaussian distributed target class.

The first class separability measure used in the B&B method was the Hellinger distance measure. Since the Hellinger distance method requires the estimation of the class PDFs, the algorithm was given a Grid Scale Factor of 1.1 and a sampling interval *dX* of 0.1. The grid scale factor was varied over a range of [1.1, 2.0] and did not affect the algorithm's results. Similarly, the sampling interval was varied between 0.0001 and 0.1 and this also did not affect the results. With the data set described above, the Hellinger branch and bound algorithm selected features three and four. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms obtained for each training set with only the selected features are shown in Figure 55.

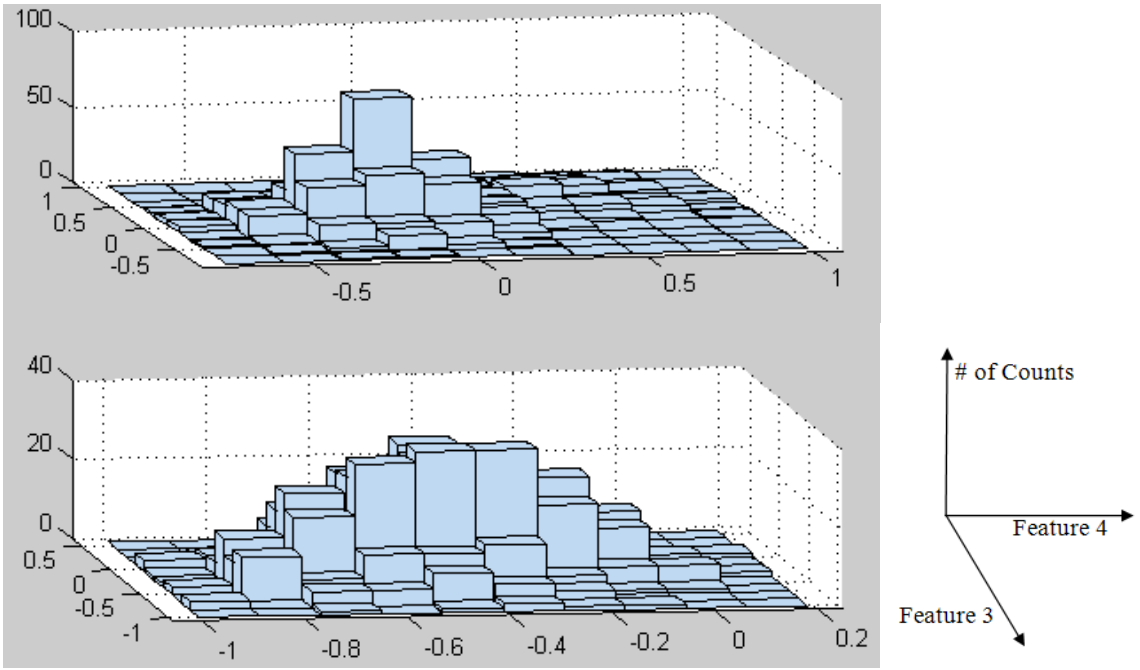

Figure 55. Two-dimensional histograms of the two classes with the selected subset of features, three and four, obtained using the B&B algorithm with the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 55 is shown as a scatter plot in Figure 56. These two figures show the separability of the two class PDFs chosen by the Hellinger B&B algorithm.

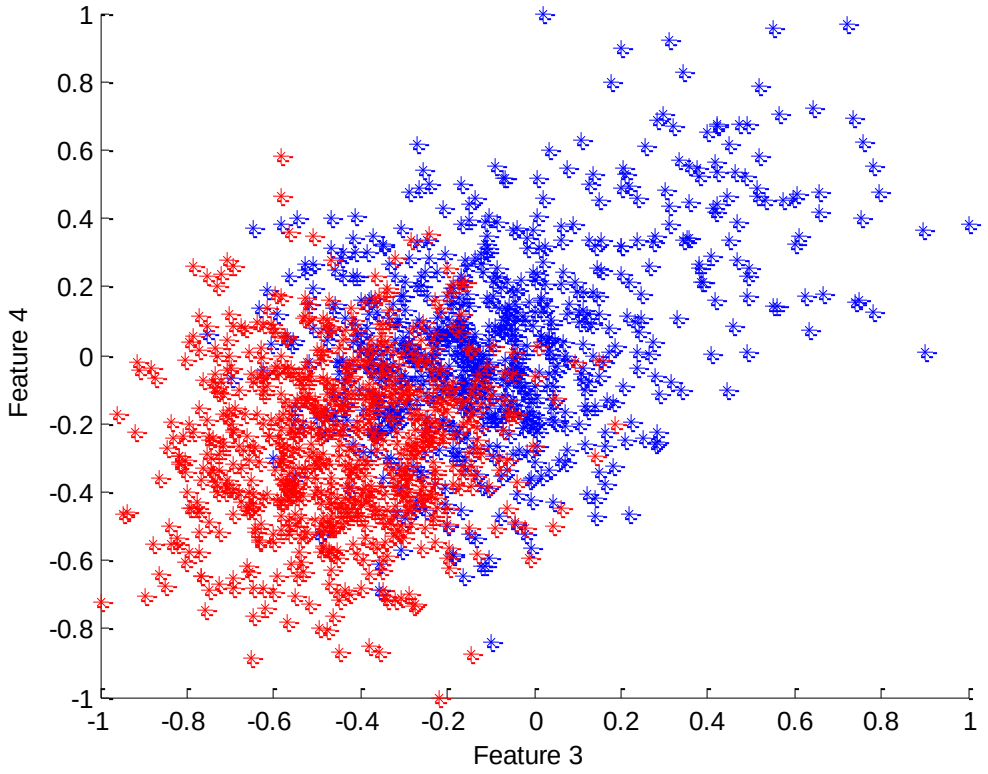

Figure 56. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 98.2% of the test data. The ROC curve for this implementation is shown in Figure 57.

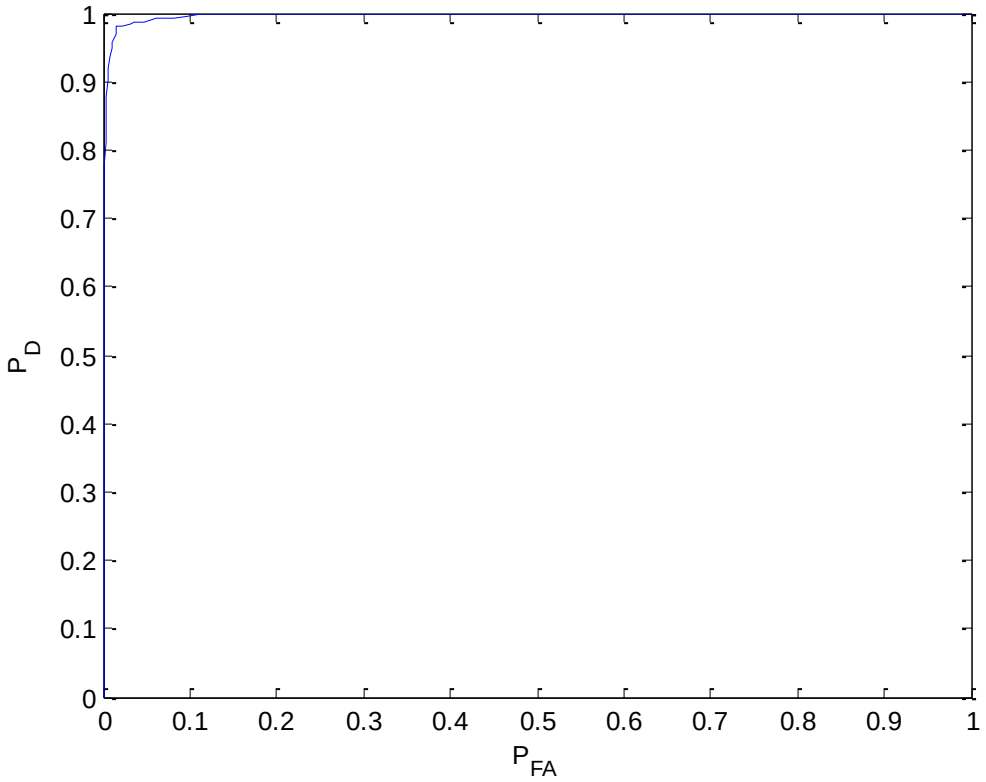

Figure 57. ROC curve obtained with the B&B algorithm and the Hellinger distance for the two non-Gaussian distributed target classes using features three and four.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 58.

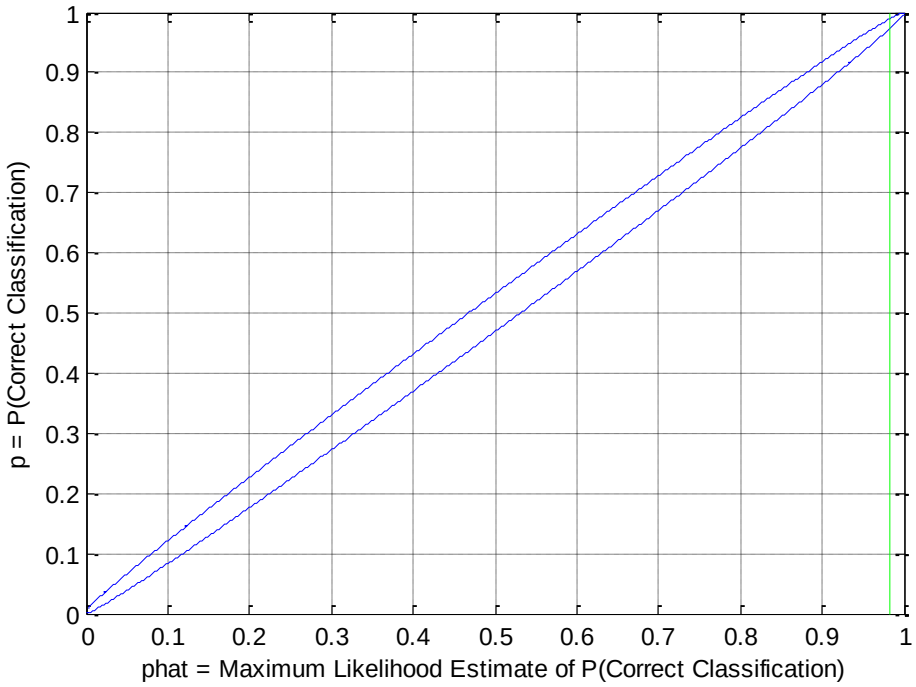

Figure 58. 95% confidence interval for the correct classification rate of 98.2% for the B&B algorithm using the Hellinger distance results with a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 97.15% and an upper bound of 98.89%. Since the B&B algorithm is a top-down selection algorithm, it is important to note the order in which features are discarded and to evaluate the changes in the Hellinger distance throughout. The features were discarded in the order [1 5 2]. The change in the Hellinger distance as features are removed is shown in Figure 59.

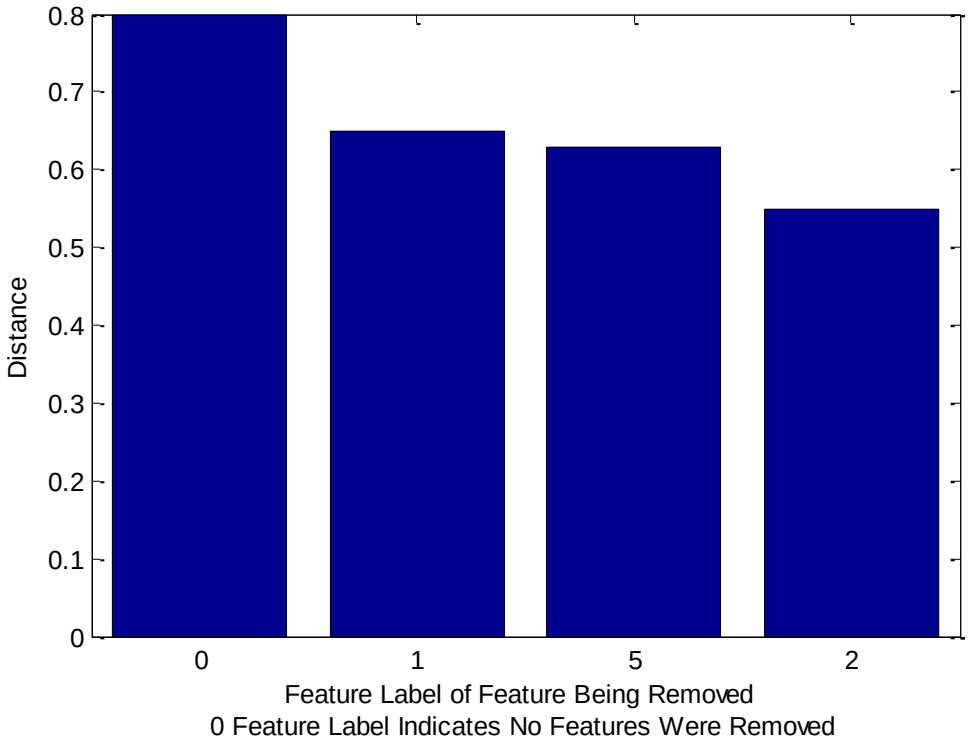

Figure 59. Changes in the Hellinger distance as features are removed from the available set.

The same data set was then evaluated using the B&B method with the Bhattacharyya distance measure as the selection criterion. The Bhattacharyya algorithm selected a different subset of features: features two and three. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms for each training set with only the selected features are shown in Figure 60.

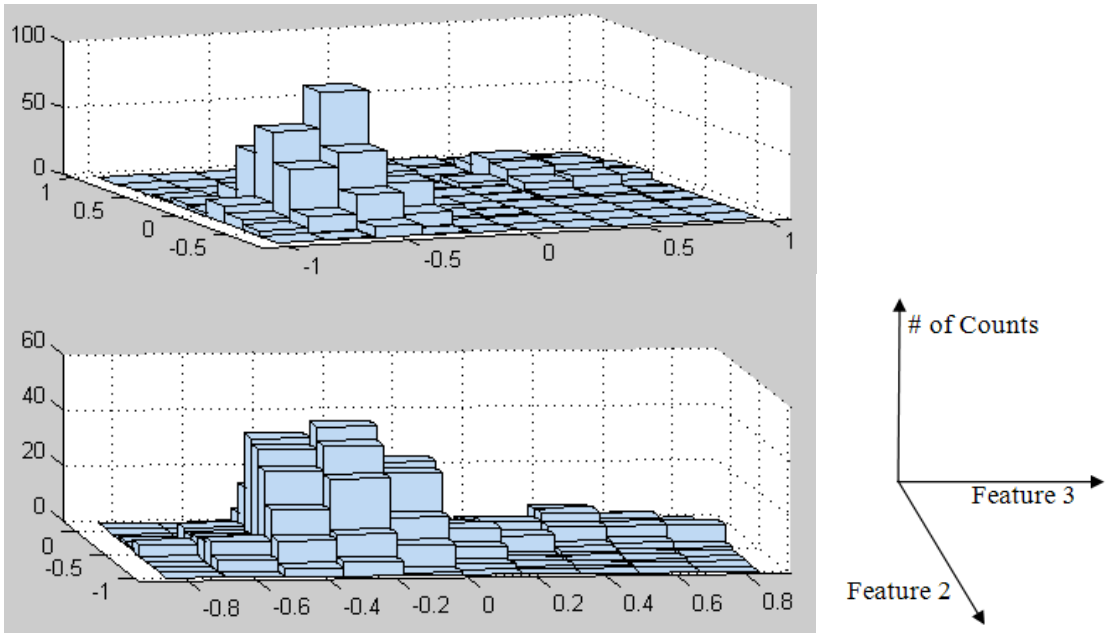

Figure 60. Two-dimensional histograms of the two classes with the selected subset of features, two and three, obtained using the B&B algorithm with the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 60 is shown as a scatter plot in Figure 61. These two figures show the separability of the two class PDFs chosen by the Bhattacharyya B&B algorithm.

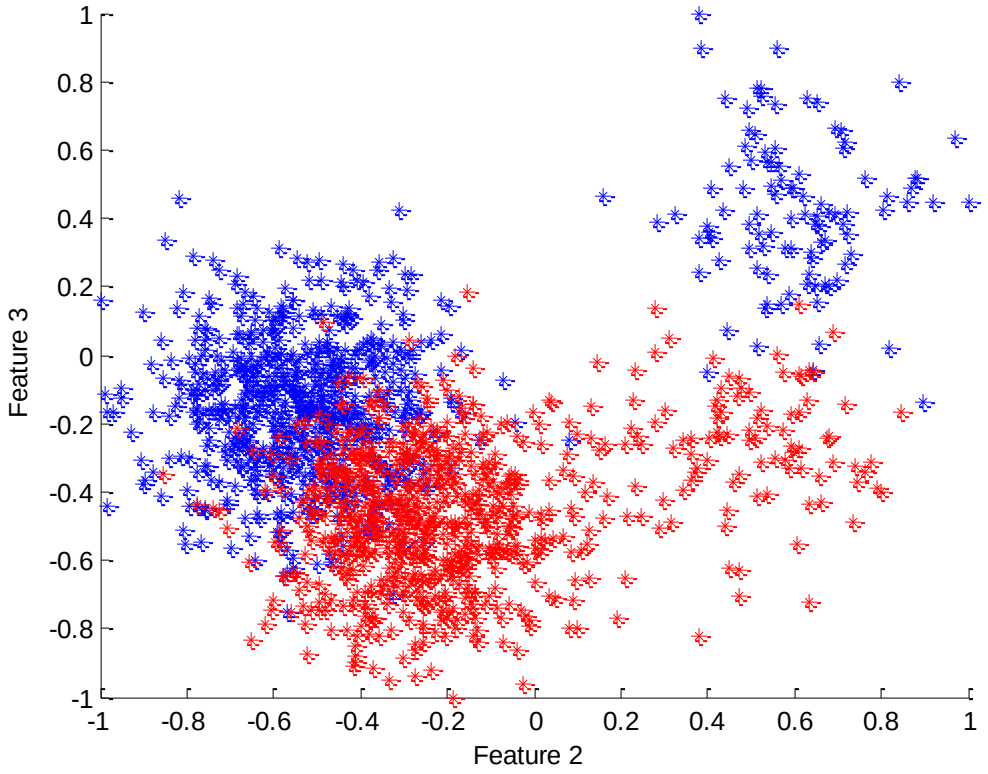

Figure 61. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 84.1% of the test data. The ROC curve for this implementation is shown in Figure 62.

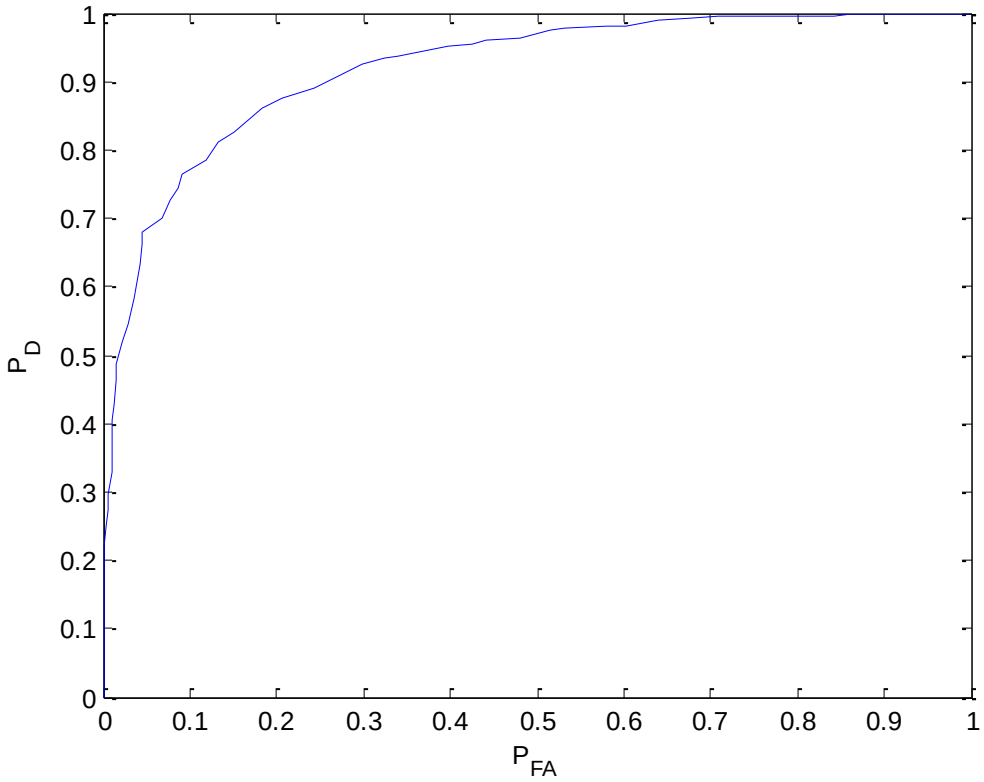

Figure 62. ROC curve obtained using the B&B algorithm with the Bhattacharyya distance for the two non-Gaussian distributed target classes using features two and three.

The next step in evaluating this algorithm's performance was to compute the 95% confidence interval for the test data given the set size and correct classification rate. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 63.

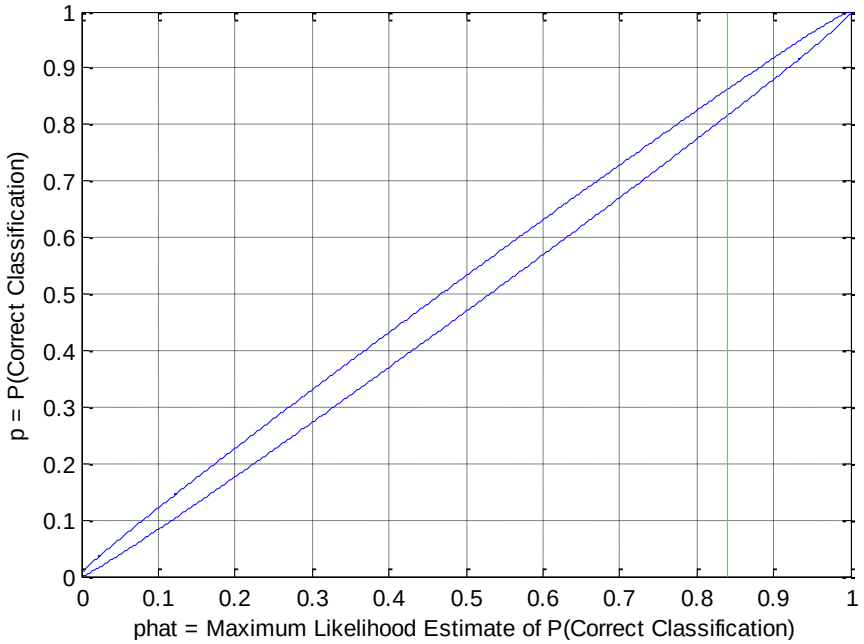

Figure 63. 95% confidence interval for the correct classification rate of 84.1% for the B&B algorithm using the Bhattacharyya distance results with a test set size of 1000 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 81.65% and an upper bound of 86.28%. Since B&B algorithm is a top-down selection algorithm, it is important to note the order in which features are discarded and to evaluate the changes in the Bhattacharyya distance throughout. The features were discarded in the order [4 5 1]. The change in the Bhattacharyya distance as features are removed is shown in Figure 64.

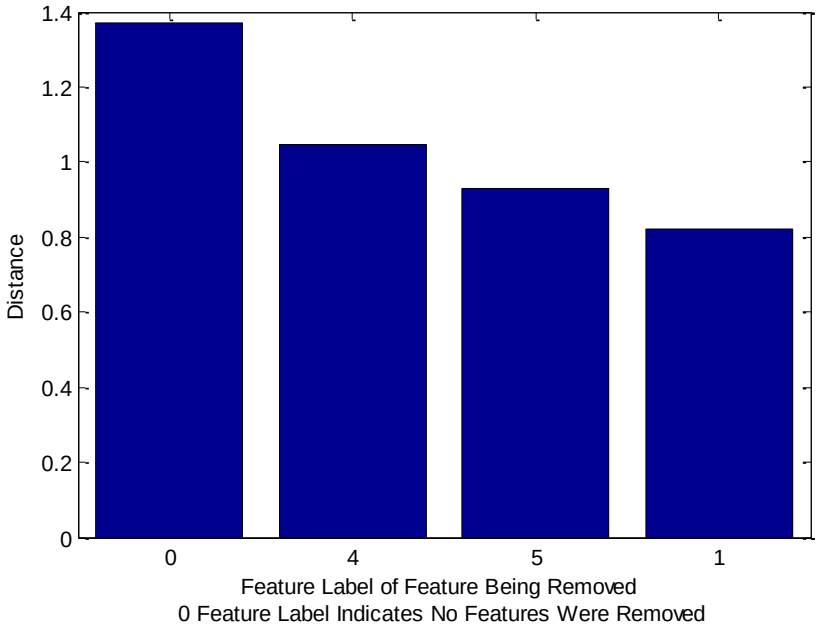

Figure 64. Changes in the Bhattacharyya distances as features are removed from the available set.

The same data set was then evaluated using the B&B method with the Mahalanobis distance measure as the selection criterion. The Mahalanobis algorithm selected the same subset of features as the Bhattacharyya algorithm, giving the same correct classification rate of 84.1%.

The exhaustive search algorithm, using each of the three class separability measures, provided the same results as the branch and bound data. This was expected due to the fact that B&B and exhaustive search are both optimal search techniques. These results, like those for the exhaustive search and the SFS, show that the Hellinger was once again most capable of selecting the best feature subset among the available feature combinations.

4. Summary

For all of the tests conducted using various distributions of classes, the Hellinger distance algorithms performed equal to or better than the other distance measures. There were several simulations performed where all algorithms selected the same subset of features. The cases previously highlighted show the simulations where the Hellinger

algorithms performed best among the three distance measures. These results bolster the notion that the removal of the Gaussian assumption from the feature selection process can greatly increase a recognition system's performance.

B. REAL DATA

In order to establish a benchmark test and create results that are comparable to other selection algorithms and criteria, it is necessary to test these algorithms with a commonly used set of classification data. For this research, the Fisher Iris data set was used. This data set consists of 50 measurements of three classes of iris flowers (Setosa, Virginica, and Versicolor). There are four features corresponding to the length and width of the sepal and petal of an iris flower. In order to continue the binary class case, only the data corresponding to the Virginica and Versicolor classes were used. These classes were chosen because MATLAB offers a two-class classification problem example using these two classes. Also, of the three classes, these two are non-linearly separable.

After the two classes were formatted as discussed in Chapter V, the distributions of each feature for each class were computed. For these simulations, the Versicolor and Virginica classes were the H_0 and H_1 classes, respectively. The classes were first separated into training and tests sets. The algorithms were trained with 60 percent of the available feature vectors, 30 vectors for each class and tested with 40 percent. The distributions for each of the classes of features are shown in Figures 65 and 66.

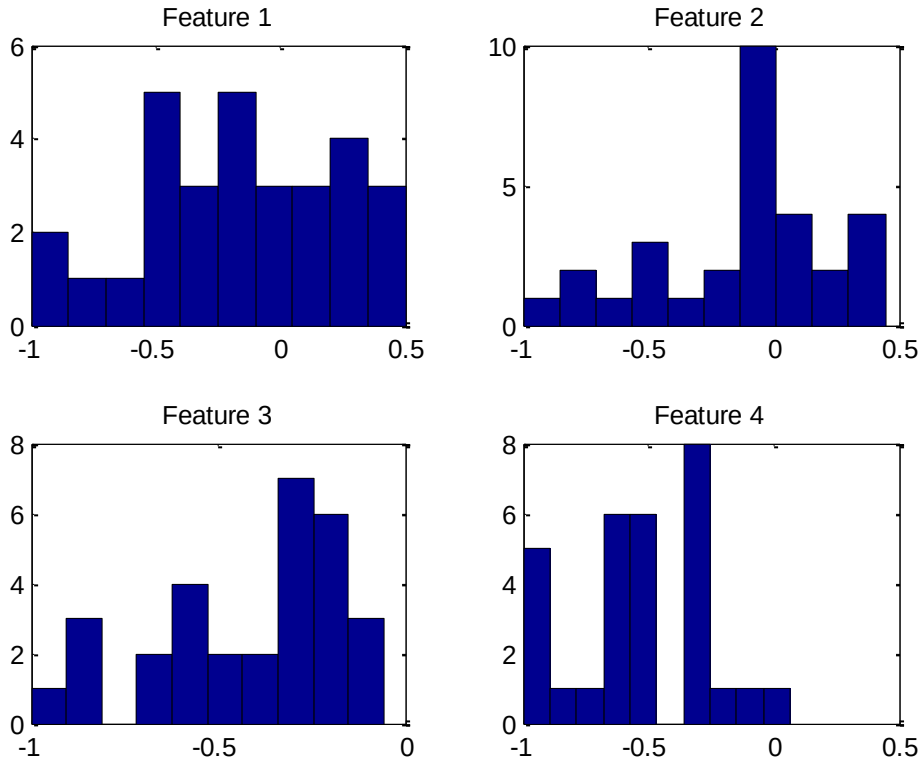

Figure 65. Feature distributions for each feature of H_0 in feature space for the Fisher iris data.

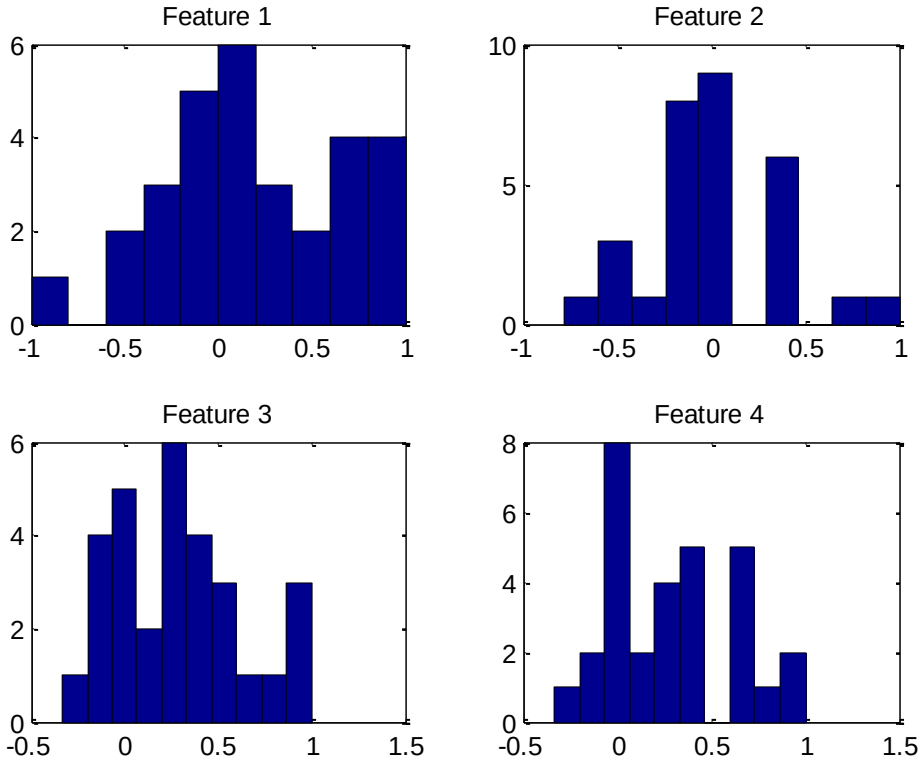

Figure 66. Feature distributions for each feature of H_1 in feature space for the Fisher iris data.

In order to properly evaluate this data set using the feature selection and classification algorithms for this research, it was important to evaluate the different sizes of feature subsets. After separating into training and test sets, the test data were classified using all four available features. The Bayes classifier constructed for this research provided a maximum correct classification rate of 95.00% for the binary class case. The ROC curve for this case is shown in Figure 67.

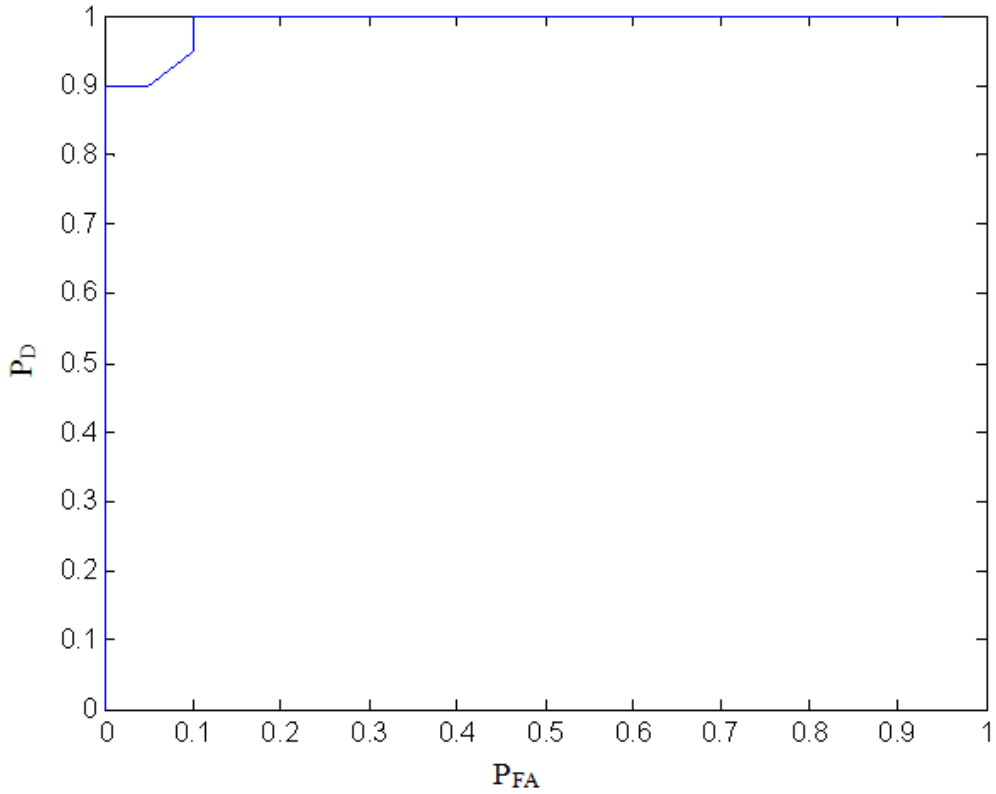

Figure 67. ROC curve for the two Fisher iris target classes using all four available features.

In order to further evaluate the classifier's performance, it is important to compute the 95% confidence interval given the correct classification rate of 95.00% and a test size of 40 vectors. The confidence interval curves and the intersection with the correct classification rate are shown in Figure 68.

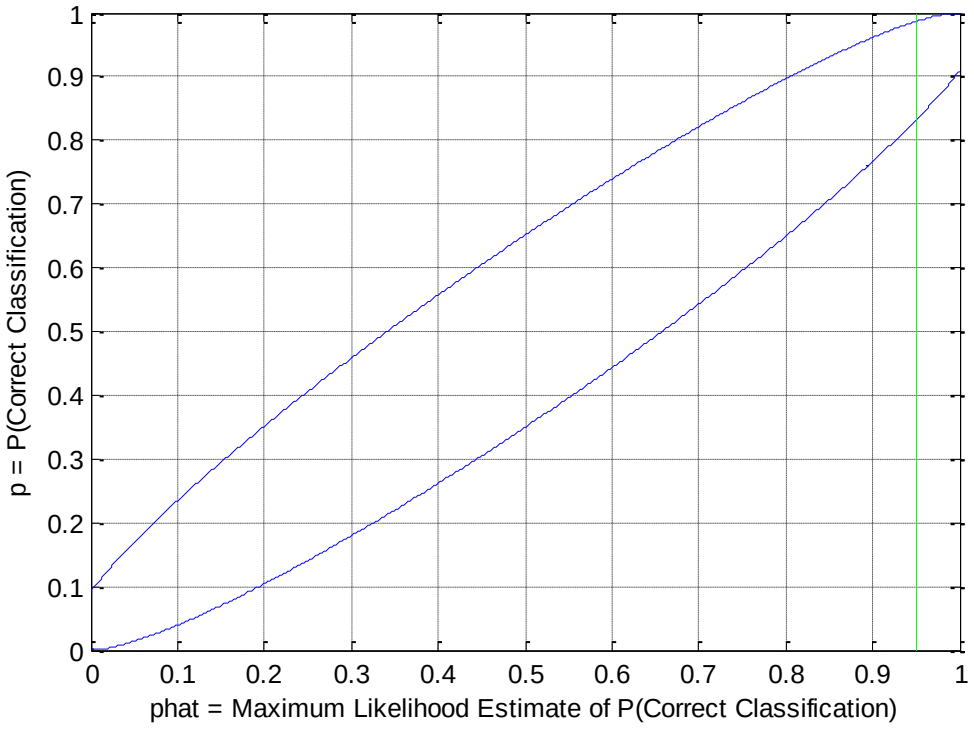

Figure 68. 95% confidence interval for the correct classification rate of 95.00% with a test set size of 40 vectors.

The evaluation of the confidence interval with these parameters yielded a lower bound of 83.17% and an upper bound of 98.65%. This interval is much larger than those for the simulated data sets due to the smaller number of test vectors.

The first feature selection test performed on the two class Fisher iris data was an exhaustive search method to choose the best two features using the Hellinger distance as the selection criterion. This algorithm selected features one and four. This information was then used to construct the final training and test vectors as discussed in Chapter V. The two-dimensional histograms obtained for each training set with only the selected features are shown in Figure 69.

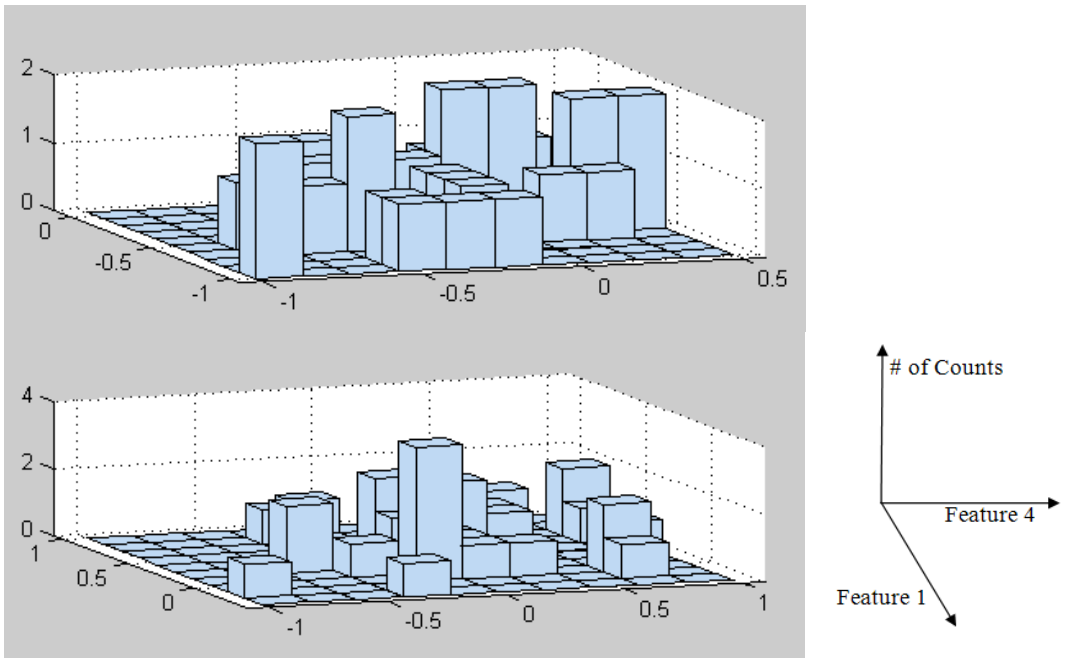

Figure 69. Two-dimensional histograms of the two classes with the selected subset of features, one and four, obtained using the exhaustive search method and the Hellinger distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The same data represented in Figure 69 (above) is shown as a scatter plot in Figure 70. These two figures show the separability of the two class PDFs chosen by the Hellinger algorithm.

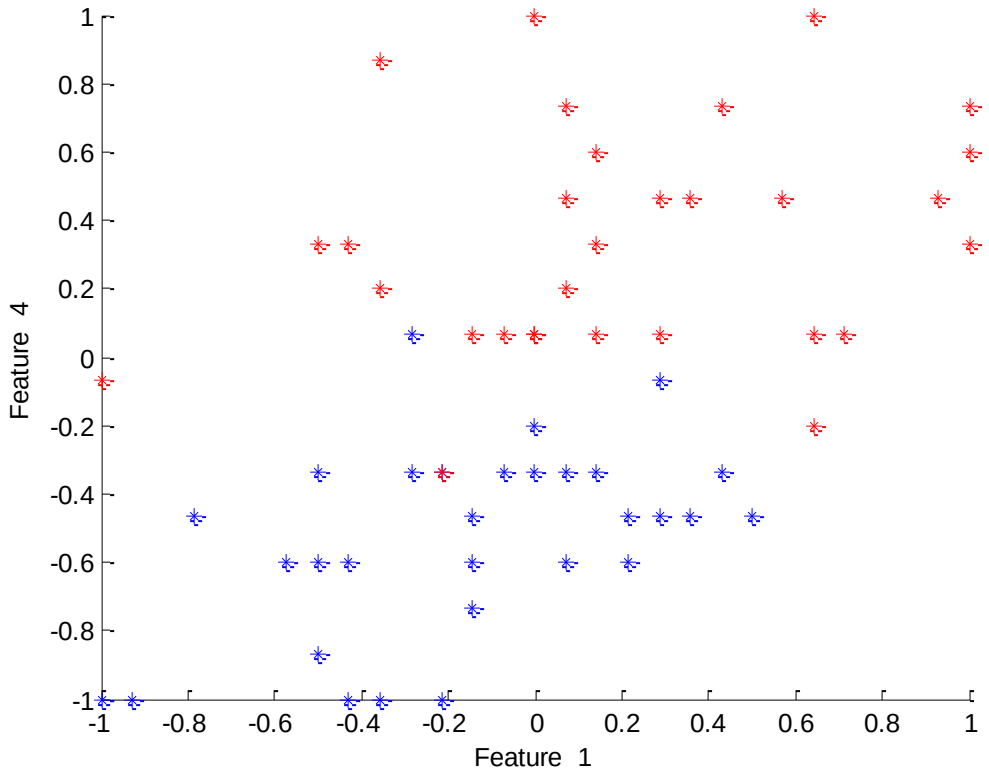

Figure 70. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 95.00% of the test data. The ROC curve for this implementation is shown in Figure 71.

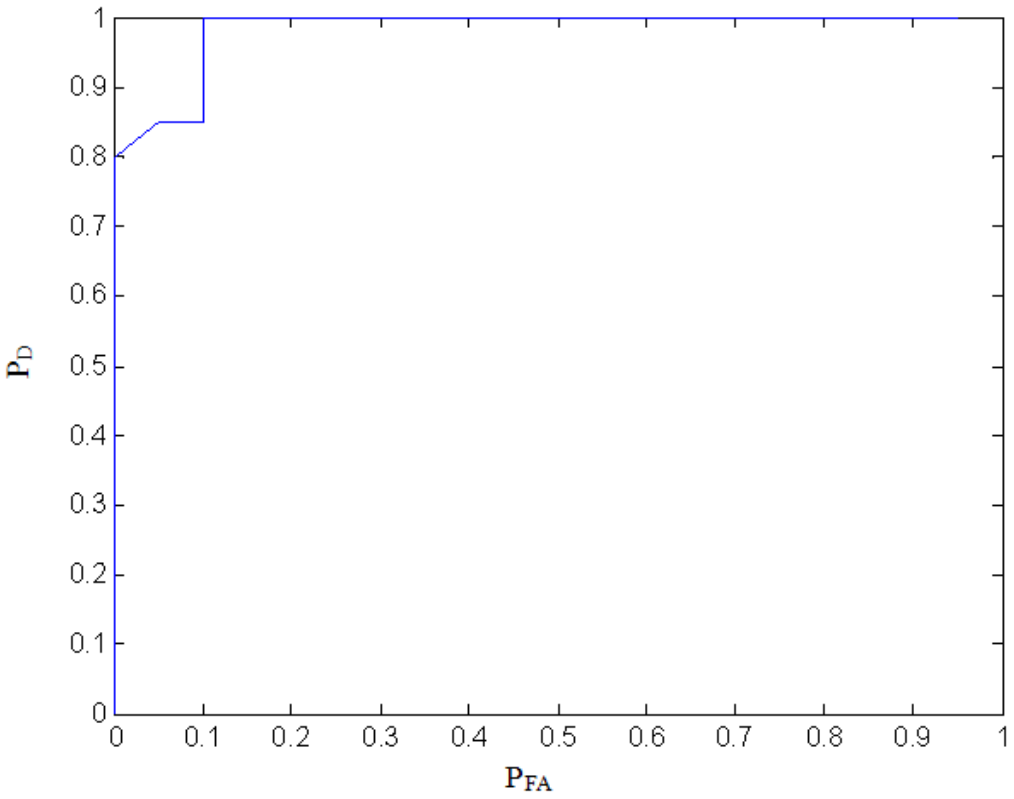

Figure 71. ROC curve obtained using the exhaustive search with the Hellinger distance for the two Fisher iris target classes using features one and four.

The 95% confidence interval for this classification rate is the same as that for the full feature set shown in Figure 68.

The same data set was then evaluated using the exhaustive search method with the Hellinger distance measure as the selection criterion in order to choose the best three feature subset. The features chosen were features two, three, and four. This subset was then used to classify the test data, and the maximum classification rate returned was 97.50%. The ROC curve for this implementation is shown in Figure 72.

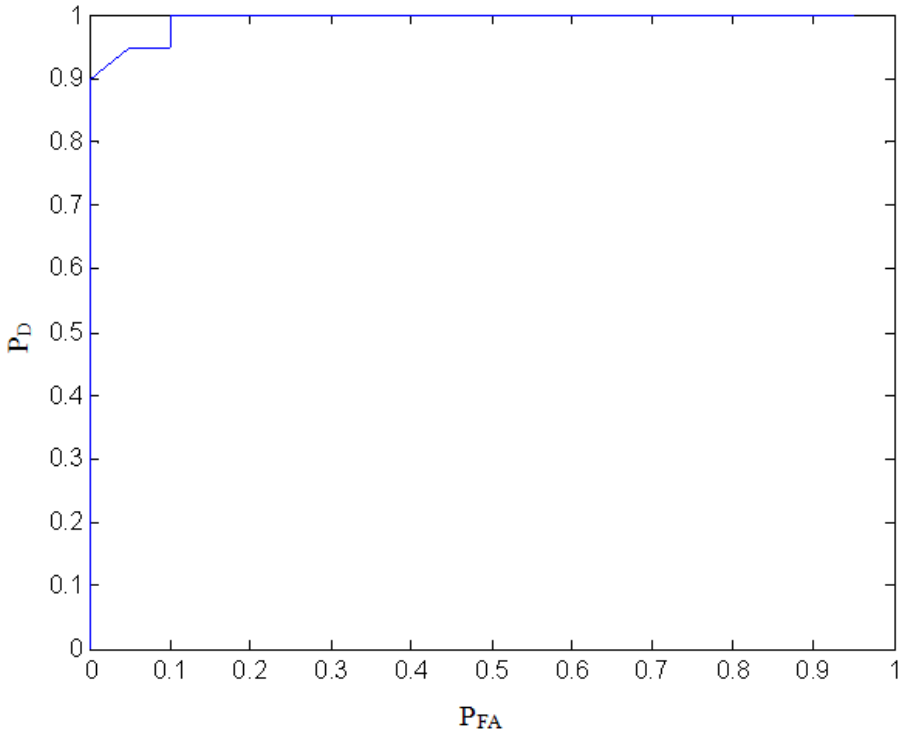

Figure 72. ROC curve obtained with the exhaustive search method and the Hellinger distance for the two Fisher iris target classes using features two, three, and four.

The next feature selection test conducted on the two class Fisher iris data was the exhaustive search algorithm using the Bhattacharyya distance as the selection criterion. The best two feature subset provided by this algorithm was features one and three. The two-dimensional histograms obtained for each training set with only the selected features are shown in Figure 73.

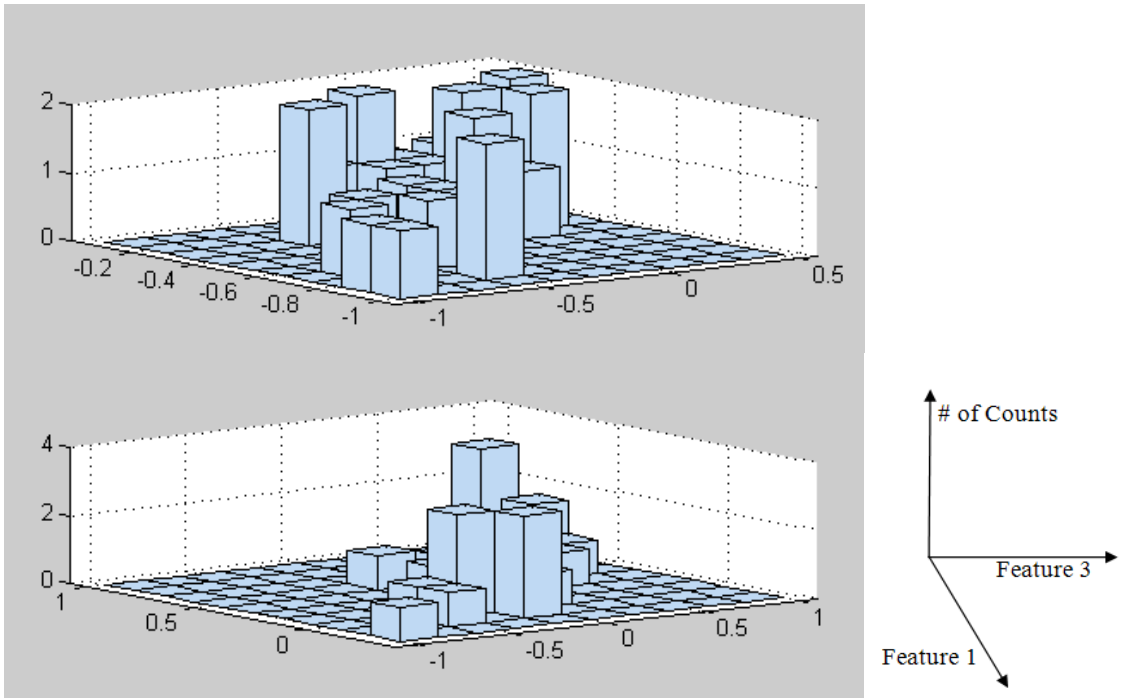

Figure 73. Two-dimensional histograms of the two classes with the selected subset of features, one and three, obtained with the exhaustive search method and the Bhattacharyya distance. Top and bottom histograms are for H_0 and H_1, respectively. The diagram on the right defines the axes for the plot.

The scatter plot for the two classes in two-dimensional feature space is shown in Figure 74.

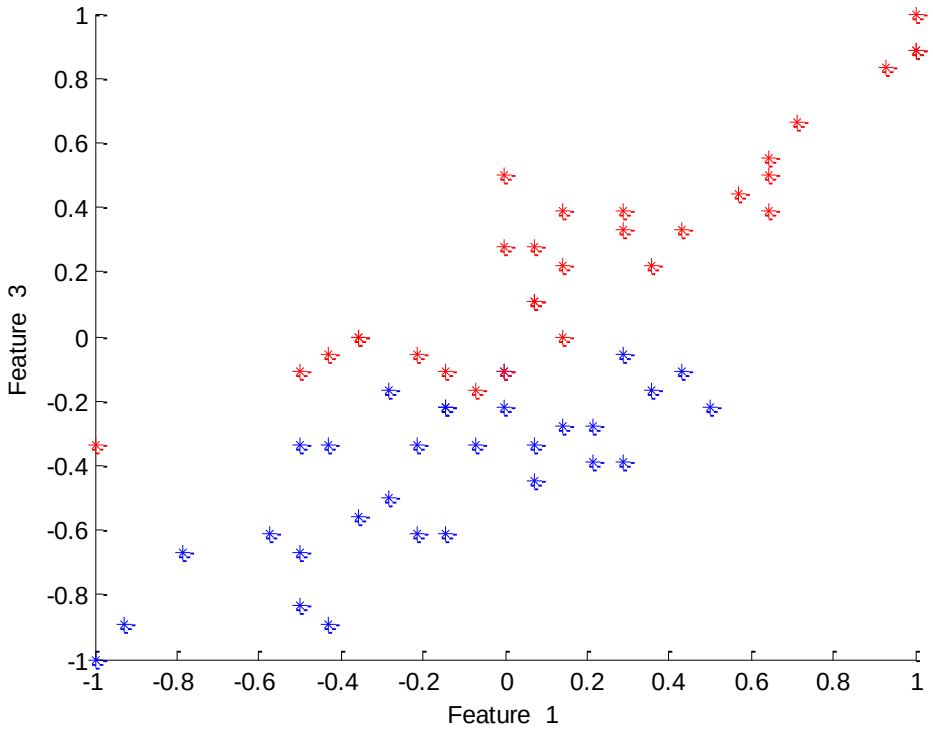

Figure 74. Scatter plot of the two-dimensional feature space for the two classes. Blue and red dots correspond to the H_0 and H_1 training data, respectively.

With this subset of features, the Bayes classifier provided a maximum correct classification rate of 95.00% of the test data. The ROC curve for this implementation is shown in Figure 75.

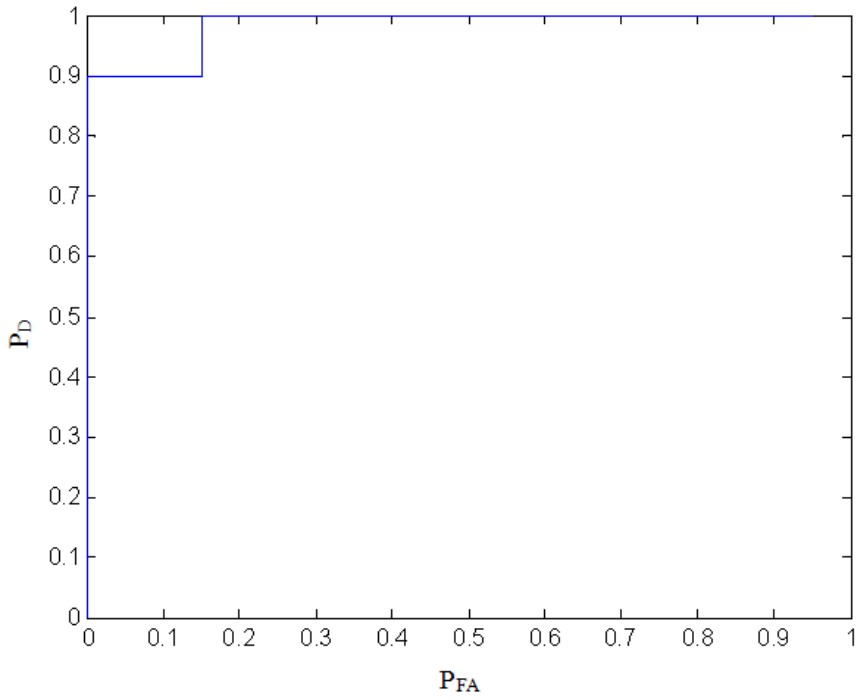

Figure 75. ROC curve obtained using the exhaustive search with the Bhattacharyya distance for the two Fisher iris target classes using features one and three is shown in this figure.

The confidence interval for this correct classification rate is the same as that obtained for the classification using all four features. The next test was to use the Bhattacharyya exhaustive search algorithm to determine the best three feature subset. The algorithm selected features one, three, and four. The correct classification rate for this feature subset was 97.50%. Despite picking a different subset of features, this subset provided the same correct classification rate and confidence interval.

The same two-class Fisher iris data were then used with the Mahalanobis exhaustive search algorithm. For the best two-feature subset, the algorithm selected features one and four, the same subset as the Hellinger algorithm, providing a correct classification rate of 95.00%. For the best three-feature subset, the algorithm chose features one, two, and three, providing a correct classification rate of 95.00%. This subset was the only three-feature subset to not show an increase in the correct classification rate.

The data were then used in the branch and bound algorithm with each of the three separability measures. The two and three feature subsets chosen for each algorithm were the same as those selected by the exhaustive search. This result is expected since both feature selection algorithms are optimal search techniques.

Overall, the results presented in this chapter do indicate an improvement in an ATR system when the feature selection process utilizes the Hellinger distance. In the next chapter, we present conclusions and discuss topics for further research.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSIONS AND RECOMMENDATIONS

The application of the Hellinger distance measure to the feature selection process was examined in this thesis. It began with an introduction to the concepts of feature selection, density estimation, and classification. The objective, as stated in Chapter I, was to determine if the Hellinger distance measure was capable of creating larger class separability than that obtained with other separability measures for the non-Gaussian data case. This hypothesis was upheld and validated by the results presented in this thesis.

When non-Gaussian target classes were introduced into the feature selection process, a significant improvement in overall correct classification was obtained when the feature subsets were chosen by the Hellinger distance. This was first presented in the case when one target class was Gaussian distributed and one class was non-Gaussian distributed. The subset chosen by the Hellinger distance provided a 7.6% increase in correct classification. Likewise, when both target classes were non-Gaussian distributed, the subset chosen by the Hellinger distance feature selection algorithm provided a 5.0% increase in correct classification.

Feature selection is most useful when the number of available features is quite large. In some instances, hundreds or thousands of features can be available. It is cost and time inefficient to use most or all of these features. The ability to reduce computational complexity, memory requirements, and time are all goals when choosing the best feature subset.

One area of further research in this area would be to extend this algorithm to a larger feature size. Due to the need to compute many multidimensional PDF estimates, this task will require efficient and creative coding techniques or a different PDF estimator. The memory constraints on many standard computers are exceeded when the dimensionality of the estimate exceeds nine or ten dimensions. This restriction forced this research to be limited to eight or fewer dimensions. Despite the obstacles, the ability to extend this algorithm to larger feature sets would constitute an important contribution to the literature.

Another area for further research is density estimation. The selection of the smoothing parameter is an important aspect of the density estimation process and improvements in this area could greatly improve the quality of the results. Also, varying the chosen kernel from the Gaussian used in this research to another might prove beneficial. An evaluation of the tradeoffs between estimate accuracy and complexity may also prove enlightening. Another idea to pursue would be to generate non-Gaussian data that are well-conditioned and have diagonal covariance matrices. This can be accomplished by using classical non-Gaussian distributions, such as the Rayleigh or uniform distributions.

APPENDIX

This is the code written for the Branch and Bound algorithm using the Hellinger distance as the selection criterion.

```
%%%%%%%%%%%%%%%%%%%%%%
%%% Wrapper for Branch and Bound Function v2
%%% Used for the Hellinger Distance Feature Selection
%%%%%%%%%%%%%%%%%%%%%%
%clear all
%close all
% load -MAT FEATURE_MATRICES_WILDER_FLIP
% load -MAT FEATURE_MATRICES_WILDER
%%%
%  X0=X0_1;      % These are used for the FLIP Matrix
%  X1=X1_1;
X0=X0_BC;
X1=X1_BC;
s=size(X0);
P=s(2);
fprintf('Number of features in a full vector=  %6.2f\n',P)
num_features=input('Enter the number of features you wish to select:')
Grid_Scale_Factor=input('Enter Grid Scale Factor for the PDF estimate: ')
dX=input('Enter the scalar value for the sampling along each dimension: ')
[dist,best,discarded_features]=Branch_and_Bound_v2(X0,X1,num_features,Grid_Scale_Factor,d
X);
dist
best
discarded_features
Grid_Scale_Factor=1.1;dX=.1;
[Grid,GridCell]=Grid_Builder_Auto_MJW_v3(X0_Train,X1_Train,Grid_Scale_Factor,dX);
f_H0=KDE_MJW(X0,GridCell);
f_H1=KDE_MJW(X1,GridCell);
f_H0_Norm=Normalize_PDF(f_H0);
f_H1_Norm=Normalize_PDF(f_H1);
classdist=Hellinger_Distance_MJW(f_H0_Norm,f_H1_Norm,dX);
fprintf('Class distance with all features is  %6.2f\n',classdist)
b=size(X0);
class1=X0;
class2=X1;
class1(:,discarded_features(1))=[];
class2(:,discarded_features(1))=[];
[Grid_New,GridCell_New]=Grid_Builder_Auto_MJW_v3(class1,class2,Grid_Scale_Factor,dX);
f_H0_New=KDE_MJW(class1,GridCell_New);
f_H1_New=KDE_MJW(class2,GridCell_New);
f_H0_New_Norm=Normalize_PDF(f_H0_New);
f_H1_New_Norm=Normalize_PDF(f_H1_New);
d(1)=Hellinger_Distance_MJW(f_H0_New_Norm,f_H1_New_Norm,dX);
rid(1)=discarded_features(1);
if length(discarded_features)>1
     for i=2:length(discarded_features)
         clear  class1  class2  Grid_New  GridCell_New  f_H0_New  f_H1_New  f_H0_New_Norm
f_H1_New_Norm
         class1=X0;
         class2=X1;
         rid(i)=discarded_features(i);
         sortrid=sort(rid,'descend');
         for j=1:length(sortrid)
             class1(:,sortrid(j))=[];
             class2(:,sortrid(j))=[];
         end

[Grid_New,GridCell_New]=Grid_Builder_Auto_MJW_v3(class1,class2,Grid_Scale_Factor,dX);
         f_H0_New=KDE_MJW(class1,GridCell_New);
         f_H1_New=KDE_MJW(class2,GridCell_New);
```

```
        f_H0_New_Norm=Normalize_PDF(f_H0_New);
        f_H1_New_Norm=Normalize_PDF(f_H1_New);
        d(i)=Hellinger_Distance_MJW(f_H0_New_Norm,f_H1_New_Norm,dX);
    end
end
dist_drop=[classdist d];
bar(dist_drop)
discarded_features1=[0 discarded_features];
for n=1:length(discarded_features1)
    Abscissa_Cell_Array_3(n) = {sprintf('%g',discarded_features1(n))};
end
set(gca,'XTickLabel',Abscissa_Cell_Array_3)
titlestr='Distance as Features are removed \n Distance with all features is %6.4f';
title(sprintf(titlestr,classdist))
ylabel('Distance')
xlabelstr='Feature Label of Feature Being Removed \n 0 Feature Label Indicates No
Features Were Removed \n Final Distance is %6.4f';
xlabel(sprintf(xlabelstr,dist))

figure
bar(diff(dist_drop))
for n=1:length(discarded_features)
    Abscissa_Cell_Array_4(n)={sprintf('%g',discarded_features(n))};
end
set(gca,'XTickLabel',Abscissa_Cell_Array_4)
titlestr1='Drop in distance as Features are removed \n Distance with all features is
%6.4f';
title(sprintf(titlestr1,classdist))
ylabel('Change in distance')
xlabelstr1='Feature Label of Feature Being Removed \n Final Distance is %6.4f';
xlabel(sprintf(xlabelstr1,dist))
```

This is the code written for the evaluation of the Hellinger distance given two PDF estimates.

```
function [hd]=Hellinger_Distance_MJW(F,G,dX)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Author: Matthew J Wilder
%%% Date:   02/23/2011
%%%
%%% This function computes the Hellinger Distance for any dimensional PDF.
%%%
%%% Inputs:     F, G          =PDF estimates- one for each class
%%%             dX            =Vector of sampling intervals for each
%%%                            feature(dimension) in the feature space
%%%                   Note:dX can be a scalar then all scaling is same
%%%
%%% Output:     hd            =scalar Hellinger distance
%%%
%%% Functions called: sumloop --> sums an array over all dimensions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=length(size(F));
if length(dX)>1
    delta_X=prod(dX);
else
    delta_X=dX^N;
end
hd = ( ( F.^(.5) - G.^(.5) ).^2 )*delta_X;
hd = sumloop(N,hd);
hd  = ( 0.5*hd )^.5;
end
```

This is the code written to compute the Bhattacharyya distance between two classes using the Gaussian assumption. This function was written by Grace A. Clark.

```
function [d] = bhattacharya_GAC(class1,class2)

% BHATTACHARYA class separation measure.
%
%   [D] = BHATTACHARYA(CLASS1,CLASS2) calculates
%   the Bhattacharya distance D between CLASS1 and
%   CLASS2.  CLASS1 and CLASS2 are M-by-P matrices
%   where M is the number of observations and P
%   is the number of features.

%
%   Author:  Dave Scott, 9/4/96 (dave-scott@llnl.gov)
%   Copyright (c) Lawrence Livermore National Laboratory
%

[num_class2] = size(class2,1);  %p
[num_class1] = size(class1,1);  %k
mu_class2 = (1/num_class2*sum(class2))';
mu_class1 = (1/num_class1*sum(class1))';
sigma_class2 = cov(class2);
sigma_class1 = cov(class1);
%sigma_class2 = mve(class2); % Robust Covariance
%sigma_class1 = mve(class1); % Robust Covariance
sum_sigma = sigma_class1 + sigma_class2;
diff_mu = mu_class2-mu_class1;
d = 1/8*diff_mu'*inv(sum_sigma/2)*(diff_mu) +...
    1/2*log(det(sum_sigma/2)/(sqrt(det(sigma_class1*sigma_class2))));
if imag(d)
    disp('Warning:  Imaginary Values for Bhattacharya Distance.')
    disp('          Results may be inaccurate.')
end
```

This is the code written to compute the Mahalanobis distance between two classes. This code was written by Kardi Teknomo.

```
function d=MahalanobisDistance(A, B)
% Return mahalanobis distance of two data matrices
% A and B (row = object, column = feature)
% @author: Kardi Teknomo
% http://people.revoledu.com/kardi/index.html
[n1, k1]=size(A);
[n2, k2]=size(B);
n=n1+n2;
if(k1~=k2)
    disp('number of columns of A and B must be the same')
else
    xDiff=mean(A)-mean(B);       % mean difference row vector
    cA=cov(A);
    cB=cov(B);
    pC=n1/n*cA+n2/n*cB;          % pooled covariance matrix
    d=sqrt(xDiff*inv(pC)*xDiff'); % mahalanobis distance
end
```

This is the code written to compute the training data PDF estimate over a specified grid.

```
function [F]=KDE_MJW(X,Grid_Cell)
%%%%%%%%%%%%%%%%%%%%%
%%%This function will computer the PDF estimate of class X over the multi-
%%%dimensional grid given in the cell array, Grid_Cell
%%%
%%%Author: Matthew Wilder
%%%
%%%Date:    03/03/2011
%%%
%%%Inputs:      X          =Class (contains measurements of features)
%%%
%%%             Grid_Cell  =Cell array containing the grid along each
%%%                         feature dimension of the class X
%%%
%%%Output:      F          = PDF estimate of X
%%%%%%%%%%%%%%%%%%%%%
sizeX=size(X);
N=sizeX(2);
%%%%The first step is to generate a combination matrix of the grid
%%%%parameters
vout=cell(size(Grid_Cell));             %This establishes the size for vout
[vout{:}]=ndgrid(Grid_Cell{:});         %vout is the multidimensional grid
%%%%%%%%%%%%%%%%%%%%%
% These two lines are to pre-establish the size of v for speed in the loop
% and are not necessary
k=size(vout);
v=zeros(numel(vout{1}),k(2));
%%%%%%%%%%%%%%%%%%%%%
for a=1:N                       %This 'for' loop creates a combination
    v(:,a)=vout{a}(:);          %matrix to allow for density estimation
end                             %for each point in every dimension
l=size(v);
%%%%%%%%
%Need to generate sigma for PDF estimation
%%%%%%%%
[sigma] = Specify_Sigma_ND_GAC_v2(X);
%%%%%%
% For KDE, put the estimates into one long vector and then use 'reshape' to
% reshape into the N-dimensional matrix
%%%%%%
f=zeros(1,l(1));
%%% This for loop can be vectorized by simply adjusting kde_One_Point
%%% to take the combination matrix, 'v'
for a=1:l(1)
    test=v(a,:)';
    %f(a)=sum(test);             %This is simply to test the reshape command
    f(a)=kde_One_Point_GAC(test,X,sigma);
end
F=reshape(f,size(vout{1}));
```

This is the code written to determine the grid of values over which to compute the training data PDF estimate.

```
Function [Grid,Grid_Cell]=Grid_Builder_Auto_MJW_v3(X_H0,X_H1,Grid_Scale_Factor,dX)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   PUPOSE:  This code automatically builds a grid of feature values over
%            which we desire to compute pdf estimates using a kernel
%            density estimator.  It finds the min and max of each feature
%            based on the measurements, then expands the min and max so we
%            can better estimate the tails of the distribution. This
%            version, v2, creates a combined grid for both classes.
%
%
%   AUTHOR:              Matthew Wilder
%   ORIGINATION DATE:    02/22/2011
%   LATEST MOD. DATE:
%
%   INPUTS:
%       X_H0,X_H1                  = M x N array of mearured feature vectors for one class
%                                M = number of measured feature training vectors
%                                N = Number of features in one training vector
%       Grid_Scale_Factor    = 1 x N vector of scale factors by which to scale the grid.
%                     The idea is to expand the range of the grids beyond the min and max
%                     values of each of the features, because we want to capture the
%                information in the tails of the distribution.  Note that each feature has
%                                its own scale factor
%                CAN BE SCALAR AND THEN ALL DIMENSIONS SCALED BY THIS VALUE
%
%       dX                   = 1 x N vector of feature sample intervals
%                                dx1,dx2, etc.
%                CAN BE SCALAR AND THEN ALL GRID DIMENSION ARE SCALED BY
%                THIS VALUE
%
%
%   OUTPUTS:
%       GRID                 = Matrix containing grid values over which we
%                                wish to compute the pdf estimates
%
%
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
% BUILD AN APPROPRIATE GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % Retrieve parameters from the feature vector data arrays X_H0 and X_H1
    % We assume that X_H0 and X_H1 have the same number of features N
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        SIZE_X_H0  = size(X_H0);
        M_H0       = SIZE_X_H0(1);    % Number of measured N x 1 training feature vectors
        N          = SIZE_X_H0(2);    % Number of features in one feature vector

        SIZE_X_H1  = size(X_H1);
        M_H1       = SIZE_X_H1(1);    % Number of measured N x 1 training feature vectors

    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % Find the min and max values of all of the features
    % in the arrays X_H0 and X_H1
    % Form a 1 x N matrix of min or max values, corresponding to each
    % feature
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            % If A is a matrix, min(A) treats the columns of A as vectors,
            % returning a row vector containing the minimum element from each column.
```

```
        X_H0_min    = min(X_H0);   % 1 x N vector of minimum values of the N features
        X_H0_max    = max(X_H0);   % 1 x N vector of maximum values of the N features

        X_H1_min    = min(X_H1);   % 1 x N vector of minimum values of the N features
        X_H1_max    = max(X_H1);   % 1 x N vector of maximum values of the N features

        X_min       = min(X_H0_min,X_H1_min);   % Min for the combined grid
        X_max       = max(X_H0_max,X_H1_max);   % Min for the combined grid

    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % Scale the Min and Max values of the feature vectors
    % to allow us to calculate values outside the min and max
    % range of the measurements - so we can capture the tails of the
    % distribution.  Choose the Grid_Scale_Factor manually
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    if length(Grid_Scale_Factor)>1
        Grid_min     = Grid_Scale_Factor .* X_min; % 1 x N vector of start values for the
grid
        Grid_max     = Grid_Scale_Factor .* X_max; % 1 x N vector of end values for the
grid
    else
        Grid_min     = Grid_Scale_Factor * X_min;
        Grid_max     = Grid_Scale_Factor * X_max;
    end

    %Set up output Matrix
        Grid=zeros(3,N);
        Grid(1,:)=Grid_min;
        Grid(3,:)=Grid_max;
    %Enter Values into Matrix --> Format is 3xN as follows:
    %    [Min_ft1         Min_ft2        ....]
    %    [#pts for PDF    #pts for PDF   ....]
    %    [Max_ft1         Max_ft2        ....]

    %Loop to enter the number of points for each dimension
    if length(dX)>1
        for i=1:N
            Grid(2,i)=ceil(-(Grid(1,i)-Grid(3,i))/dX(i));
        end
    else
        for i=1:N
            Grid(2,i)=ceil(-(Grid(1,i)-Grid(3,i))/dX);
        end
    end
    if length(dX)>1
        for i=1:N
            eval(['GridX' num2str(i) ' =Grid(1,i):dX(i):Grid(3,i);']);
        end
    else
        for i=1:N
            eval(['GridX' num2str(i) ' =Grid(1,i):dX:Grid(3,i);']);
        end
    end
    for a=1:N
        eval(['Grid_Cell{a}=GridX' num2str(a) ';']);
    end
end
%
%
%End of Code
```

This code was written to evaluate the PDF estimate for one given point on a grid or test point. This function was written by Grace A. Clark.

```
function [f] = kde_One_Point_GAC(X_test,X,sigma)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   PUPOSE:    This code implements a 2D kernel density estimate for a single
%               point on the pdf.  This function must be called once for
%               every point to be estimated
%
%   AUTHOR:             Grace A. Clark
%   ORIGINATION DATE:   11/11/10
%   LATEST MOD. DATE:   12/29/10
%   INPUTS:
%       X_test  = (N x 1) array of grid values for the feature space,
%                   where N = the number of features in the desired feature
%                   space (the size of the grid)
%                   Our goal is to estimate the pdf values at all the
%                   points on the grid
%
%       X       = (M x N) array of M feature vectors (rows), each having
%                   N features (elements)
%
%       sigma   = Scalar smoothing parameter for the kernel density
%                   estimator
%   OUTPUTS:
%       f       = kde estimate at one point on the grid
%   FUNCTIONS CALLED:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% RETRIEVE NECESSARY PARAMETERS FROM THE DATA MATRIX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    SIZE_X  = size(X);   % Get the matrix dimensions
    M       = SIZE_X(1);    % Number of training feature vector
    N       = SIZE_X(2);    % Number of features in one feature vector

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    f_arg = 0;      % Initialize

    Q = 1/( (2*pi)^(N/2)*(sigma^N)*M );     % The scale constant for the sum

    for m = 1:M

        X_train = X(m,:)';  % Take the transpose to get the right dimensionality for the
                            % quadratic form in f_arg below.
                            % I want an N x 1 vector

        f_arg   = f_arg + exp(-( (X_test - X_train)'*(X_test - X_train) )/2*sigma );

    end

        f = Q*f_arg;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% End of Main Code  %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

This code was written for the exhaustive search method using the Hellinger distance as the selection criterion.

```
function [best]=Exhaustive_Search(X0,X1)
%%%
%%% This function finds the best feature subset using the Exhaustive
%%% Search with the Hellinger distance
n1=size(X0);
n=n1(2);
k=input('Enter the number of features in the final subset: ')
GridScaleFactor=input('Enter the Grid Scale Factor: ')
dX=input('Enter the sampling interval along each dimension: ')
C=combnk(1:n,k);
sizeC1=size(C);
sizeC=sizeC1(1);    %Number of combinations
check_dist=1:sizeC;
for i=1:sizeC
    clear f* class* Grid GridCell
    for q=1:k
            eval(['f' num2str(q) ' =C(i,q);']);
    end
    for w=1:k
            eval(['class1(:,' num2str(w) ') =X0(:,f' num2str(w) ');']);
    end
    for r=1:k
            eval(['class2(:,' num2str(r) ') =X1(:,f' num2str(r) ');']);
    end
    [Grid,GridCell]=Grid_Builder_Auto_MJW_v3(class1,class2,GridScaleFactor,dX);
    f_class1=KDE_MJW(class1,GridCell);
    f_class2=KDE_MJW(class2,GridCell);
    f_class1N=Normalize_PDF(f_class1);
    f_class2N=Normalize_PDF(f_class2);
    check_dist(i)=Hellinger_Distance_MJW(f_class1N,f_class2N,dX);
    val=find(check_dist==max(check_dist));
    best=C(val,:);
end
```

This is the code written for the SFS algorithm using the Hellinger distance as the selection criterion.

```
function [Best]=SequentialForwardSelection_Hellinger(class1,class2,NumFeatComb)
%%% SFS Algorithm---Best is the vector of the best subset
GSF=1.1;
dX=.1;
NofFeatures=size(class1,2);
cLBest=[];
k=1;
while k<=NumFeatComb
    maxJ=0;
    for i=1:NofFeatures
        clear f_* Grid*
        if isempty(find(cLBest==i))
            combi=[cLBest i];
        else continue;
        end
        [Grid,GridCell]=Grid_Builder_Auto_MJW_v3(class1(:,combi),class2(:,combi),GSF,dX);
        f_class1=KDE_MJW(class1(:,combi),GridCell);
        f_class2=KDE_MJW(class2(:,combi),GridCell);
        f_class1N=Normalize_PDF(f_class1);
        f_class2N=Normalize_PDF(f_class2);
        J=Hellinger_Distance_MJW(f_class1N,f_class2N,dX);
        if J>maxJ
            maxJ=J;
            sofar=combi;
        end
    end
    Best=sort(sofar,'ascend');
    k=k+1;
end
```

This is the code written for the Cain algorithm for automatic selection of the PDF estimate smoothing parameter.

```
function    sigma = sigma_design(vectors_in_class)
% ----------------------------------------------------------
%  Title:   sigma_design
%  Author:  Peter Cheng
%  Date:    July/96
%  Purpose:   This code is to determine the optimum value of
%             sigma for distinct class. This algorithm can be
%             referenced in "An improved PNN and its performance
%             relative to other models" by J.Bibb Cain.
%         This appeared in SPIE vol.1294 p.354  (1990)
%  Inputs:    vectors_in_class = the pattern vectors in the same
%                           class, the number of columns represents
%                           the number of features.
%  Output:     sigma = the optimum sigma in this class is equal to
%                    gain constant times average minimum distance
%------------------------------------------------------------------
[nr,nc]=size(vectors_in_class);
dist = zeros(1,nr);
for  ii=1:nr;
       tmp_transpose = vectors_in_class(ii,:)';
       tmp_repeat = tmp_transpose(:,ones(1,nr));
       tmp_repeat = tmp_repeat' ;
       vector_diff = vectors_in_class - tmp_repeat;
       vector_dist = vector_diff.* vector_diff;
       if nc > 1
        dist_sq = sum(vector_dist')' ;
       else
         dist_sq = vector_dist;
       end
       [val,ind] = sort(dist_sq) ;
       dist(ii) = sqrt(val(2)) ;     % the first one is zero
 end
 g_gain = 1.3;
 sigma = g_gain * sum(dist)/nr ;
```

This code was written to compute the 95% confidence intervals and the images presented in this thesis. This code was written by Grace A. Clark.

```
function [N,PHAT,LLLe,UUUe]  = Conf_Int_Binomial_GAC(N,PHAT)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%  M-File   Conf_Int_GAC_v1.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%
%   This code compute statistical confidence intervals for the
%   probability of correct classification P(CC) of a detector, given
%   the sample size, n.
%   We assume a binary detection problem with hypotheses H0 and H1
%   The intervals are about a binomial distribution
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%
%   Author:                 Grace A. Clark
%
%   Date Originated:        10/8/07
%   Date Last Modified:     11/7/07
%
%   Source Code:            Heart Valve and Shell Oil project codes I wrote
%                           years ago  1/18/96
%
%
%   Inputs:
%       N           = (Scalar) Integer number of samples in the statistical test
%                   = No. H0 samples + No. H1 samples
```

```
%       phat        = (Scalar) maximum likelihood estimate of the P(CC)
%                       calculated empirically elsewhere from a confusion matrix and/or
%                       ROC Curve
%       deltaphat   = Discretization step in phat for making plots
%
%   Outputs:
%       Ln, Un      = Normal approximation estimates of the lower and upper
%                       bounds of the confidence interval
%       Le, Ue      = "Exact" estimates of the lower and upper
%                       bounds of the confidence interval
%
%----------------------------------------------------------------
% Define Variables Internal to the Code:
%
% Let phat =  Max. Lik. estimate of P(correct classification)
%          = a row vector (1 X Npoints)
% Let qhat = Max. Lik. estimate of P(incorrect classification) = 1 - phat
%          = a row vector (1 X Npoints)
% Let n = number of training samples or trials
%          = a row vector (1 X Nplots)
% Let sigma = variance to use in the confidence intervals
%          = a matrix (Npoints X Nplots)
% Let U    = Approximate upper bound
%          = a matrix (Npoints X Nplots)
% Let L    = Approximate lower bound
%          = a matrix (Npoints X Nplots)
% Let Ue   = Exact upper bound
%          = a matrix (Npoints X Nplots)
% Let Le   = Exact lower bound
%          = a matrix (Npoints X Nplots)
% Let Npoints = number of samples to give phat
% Let Nplots  = number of values of n to use
%----------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% Calculate the "Exact" Bounds, Given a single value of N and a single
% value of phat
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % CALCULATE THE DESIRED EXACT BOUNDS FOR GIVEN N AND PHAT
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        LLLe= ( N*PHAT + 2 - 2*(N*PHAT*(1-PHAT) +1)^.5 )/(N+4);
        UUUe= ( N*PHAT + 2 + 2*(N*PHAT*(1-PHAT) +1)^.5 )/(N+4);

            if LLLe < 0;        % Constrain the values
                LLLe = 0.;
            end

            if UUUe > 1.;
                UUUe = 1.;
            end

    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    % CALCULATE THE NORMAL BOUNDS FOR GIVEN N AND PHAT
    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

        SIGMA = ((PHAT*(1-PHAT))/N)^.5;
        LNORM = PHAT - 1.96*SIGMA;
        UNORM = PHAT + 1.96*SIGMA;

            if LNORM < 0;        % Constrain the values
                LMORM = 0.;
            end
```

```
            if UNORM > 1.;
                UNORM = 1.;
            end

    %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        % DISPLAY
        %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            disp(' ')
            disp('
########################################################################')
            disp('    #####
#####')
            disp('    #####                  "EXACT" APPROXIMATION
#####')
            disp('    #####                  95% CONFIDENCE INTERVAL
#####')
            disp('    #####
#####')
            disp('
########################################################################')
            disp(' ')
            disp(' ')


            fprintf('\t  GIVEN: %s')
            fprintf('\n')
            fprintf('\t\t  N = No. Samples         = %g\n',N)
            fprintf('\t\t  phat                    = %g\n',PHAT)
            fprintf('\n')
            fprintf('\t  95%% CONFIDENCE INTERVAL BOUNDS: %s')
            fprintf('\n')
            fprintf('\t\t  Lower Bound (Exact)     = %g\n',LLLe)
            fprintf('\t\t  Upper Bound (Exact)     = %g\n',UUUe)
            fprintf('\n')
            fprintf('\n')

            disp('
########################################################################')
            disp(' ')
            %disp(' ')
        %--------------------------------------------


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Define Plotting Parameters, Given the Inputs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        deltaphat = .001;              % Discretization size for the bound plots
        Npoints = 1.0/deltaphat + 1;   % Number of points in a plot of the bounds
        n  = [N];                      % Let the n vector be the scalar N
        Nplots = length(n);            % Number of values of n, the sample size, to use
in making the plots


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Compute the Lower and Upper Bounds on phat for the given N and PHAT
    % For BOTH the Normal and "Exact" Approximations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        phat = 0.0:deltaphat:1.0;
        qhat = 1-phat;
```

```
        for i=1:Npoints
                    for j=1:Nplots

                        % Rough Approximation of Bounds
                            sigmas(i,j) = ((phat(i).*qhat(i))./n(j)).^.5;
                            %sigma(i,j) = 1/(i+j-1);  % I used this to debug the code
                            Ls(i,j) = phat(i) - 1.96*sigmas(i,j);
                            Us(i,j) = phat(i) + 1.96*sigmas(i,j);

                        % Better (exact) Approximation of Bounds
                            Les(i,j) = ( n(j).*phat(i) + 2 - ...
                                2.*(n(j).*phat(i).*(1-phat(i)) +1).^.5 )/(n(j)+4);
                            Ues(i,j) = ( n(j).*phat(i) + 2 + ...
                                2.*(n(j).*phat(i).*(1-phat(i)) +1).^.5 )/(n(j)+4);

                                    %Constrain the confidence intervals
                                        if  Ls(i,j) < 0;
                                                        Ls(i,j) = 0;
                                        end
                                        if Us(i,j) >1.0;
                                                    Us(i,j) =1.0;
                                        end
                                        if  Les(i,j) < 0;
                                                        Les(i,j) = 0;
                                        end
                                        if Ues(i,j) >1.0;
                                                    Ues(i,j) =1.0;
                                        end
                    end
        end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Plot the Exact Bounds on a Single Graph
    % OVERLAY A VERTICAL LINE FOR THE SPECIFIC PHAT specified above

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        figure  % Start a new frame

        plot(phat,Les(:,1),'-b',phat,Ues(:,1),'-b');
        Text_CI_1 = '95%% Conf. Ints. About P(CC), Exact for Small (or Any) n';
        Text_CI_2 = ' \n For Specific PHAT = %3.3g, \t N = %3.3g';
        Text_CI_3 = ' \n Les = %1.2g, \t Ues = %1.2g, \t Green Vertical Line Marks
PHAT';
        Text_CI   = [Text_CI_1,Text_CI_2,Text_CI_3];
        title(sprintf(Text_CI,PHAT,N,LLLe,UUUe))

        hold all        % So we can plot a vertical line over the plot

        plot([PHAT,PHAT],ylim,'g');      % Plot a vertical line at PHAT (From KAW)
        % legend('Blue = Exact for Small (or Any)  n','Location','NorthWest')
        xlabel('phat = Maximum Likelihood Estimate of P(Correct Classification)')
        ylabel('p = P(Correct Classification)')
        grid

        hold off
      %-------------------------------------------------------------
        %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        % Pause and alert the user to start working again
        %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            disp(' ')
            disp('
##########################################################################')
            disp('   #####
#####')
```

```
                disp('    #####                  PRESS RETURN TO CONTINUE
#####')
                disp('    #####
#####')
                disp('
##########################################################################')
                pause;
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Plot the Exact Bounds and the Normal Approx. Bounds (for Large n) on a Single Graph
    % OVERLAY A VERTICAL LINE FOR THE SPECIFIC PHAT specifiec above

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

            figure  % Start a new frame

            plot(phat,Ls(:,1),'-b',phat,Les(:,1),'-r',phat,Us(:,1),'-b',phat,Ues(:,1),'-
r');
            Text_CI_1 = '95%% Conf. Ints. About P(CC), Normal Approx. and Exact for Small
(or Any) n';
            Text_CI_2 = ' \n For Specific PHAT = %3.3g,\t N = %3.3g';
            Text_CI_3 = ' \n Les = %1.2g, \t Ues = %1.2g, \t Ls = %1.2g, \t Us = %1.2g,
\t Green Vertical Line Marks PHAT';
            Text_CI   = [Text_CI_1,Text_CI_2,Text_CI_3];
            title(sprintf(Text_CI,PHAT,N,LLLe,UUUe,LNORM,UNORM))

            hold all        % So we can plot a vertical line over the plot

            plot([PHAT,PHAT],ylim,'g');      % Plot a vertical line at PHAT (From KAW)
            legend('Blue = Normal Approx. for Large n','Red = Exact for Small (or Any)
n','Location','NorthWest')
            xlabel('phat = Maximum Likelihood Estimate of P(Correct Classification)')
            ylabel('p = P(Correct Classification)')
            grid

            hold off
         %------------------------------------------------------------
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            % Pause and alert the user to start working again
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                disp(' ')
                disp('
##########################################################################')
                disp('    #####
#####')
                disp('    #####                  PRESS RETURN TO CONTINUE
#####')
                disp('    #####
#####')
                disp('
##########################################################################')
                pause;
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% ASK THE USER IF SHE/HE WANTS TO MAKE GENERAL PLOTS OF THE BOUNDS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

    reply_plots   = input('\n\n Make bound plots for n = [Smallest,Medium,Largest]? (y or
n) > ','s');

            if reply_plots == 'y'
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Define Plotting Parameters, Given the Inputs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        Npoints = 1.0/deltaphat + 1;     % Number of points in a plot of the bounds
        n   = [N 100 1000];              % The three sample sizes at which to plot the
conf. ints.
        Nplots = length(n);              % Number of values of n, the sample size, to use
in making the plots



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Compute the Lower and Upper Bounds on phat for Plotting Purposes
    % For BOTH the Normal and "Exact" Approximations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        phat = 0.0:deltaphat:1.0;
        qhat = 1-phat;

            for i=1:Npoints
                        for j=1:Nplots

                            % Rough Approximation of Bounds
                                sigma(i,j) = ((phat(i).*qhat(i))./n(j)).^.5;
                                %sigma(i,j) = 1/(i+j-1);  % I used this to debug the code
                                L(i,j) = phat(i) - 1.96*sigma(i,j);
                                U(i,j) = phat(i) + 1.96*sigma(i,j);

                            % Better (exact) Approximation of Bounds
                                Le(i,j) = ( n(j).*phat(i) + 2 - ...
                                    2.*(n(j).*phat(i).*(1-phat(i)) +1).^.5 )/(n(j)+4);
                                Ue(i,j) = ( n(j).*phat(i) + 2 + ...
                                    2.*(n(j).*phat(i).*(1-phat(i)) +1).^.5 )/(n(j)+4);

                                        %Constrain the confidence intervals
                                            if  L(i,j) < 0;
                                                        L(i,j) = 0;
                                            end
                                            if U(i,j) >1.0;
                                                    U(i,j) =1.0;
                                            end
                                            if  Le(i,j) < 0;
                                                        Le(i,j) = 0;
                                            end
                                            if Ue(i,j) >1.0;
                                                    Ue(i,j) =1.0;
                                            end
                        end
            end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Plot the Exact Bounds on a Single Graph

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

            figure
            plot(phat,Le(:,1),'-r', phat,Le(:,2),'-g', phat,Le(:,3),'-b',  ...
```

```
            phat,Ue(:,1),'-r', phat,Ue(:,2),'-g', phat,Ue(:,3),'-b' );
            axis('square')
            Text_Exact_1 = '95%% Conf. Ints. for P(CC) \t (Exact)';
            Text_Exact_2 = '\n Red: n = %3i,\t Green: n = %3i, \t Blue: n = %3i';
            Text_Exact = [Text_Exact_1 Text_Exact_2];
            title(sprintf(Text_Exact,n(1),n(2),n(3)))
            legend('red: n = Smallest', 'green: n = Medium', 'blue: n =
Largest','Location','NorthWest')
            xlabel('phat = Maximum Likelihood Estimate of P(Correct Classification)')
            ylabel('p = P(Correct Classification)')
            grid

         %------------------------------------------------------------
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            % Pause and alert the user to start working again
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                disp(' ')
                disp('
##########################################################################')
                disp('   #####
#####')
                disp('   #####                  PRESS RETURN TO CONTINUE
#####')
                disp('   #####
#####')
                disp('
##########################################################################')
                pause;
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Plot the Normal Approximation Bounds for Large n on a Single Graph

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        figure
        clear graph
        axis('square')
        plot(121)
        plot(phat,L(:,1),'-r',   phat,L(:,2),'-g',   phat,L(:,3),'-b',  ...
                phat,U(:,1),'-r',   phat,U(:,2),'-g',   phat,U(:,3),'-b' );
            axis('square')
            Text_Normal_1 = '95%% Conf. Ints. for P(CC) \t (Normal Approximation)';
            Text_Normal_2 = '\n Red: n = %3i,\t Green: n = %3i, \t Blue: n = %3i';
            Text_Normal = [Text_Normal_1 Text_Normal_2];
            title(sprintf(Text_Normal,n(1),n(2),n(3)))
            legend('red: n = Smallest', 'green: n = Medium', 'blue: n =
Largest','Location','NorthWest')
            xlabel('phat = Maximum Likelihood Estimate of P(Correct Classification)')
            %label_phat = '$\widehat{p}$';  % Try using LaTeX
            %label_phat = '$\frac{1}{2}$';  % Try using LaTeX
            %xlabel(label_phat)             % Try using LaTeX
            ylabel('p = P(Correct Classification)')
            grid

            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
            % Pause and alert the user to start working again
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                disp(' ')
                disp('
##########################################################################')
                disp('   #####
#####')
                disp('   #####                  PRESS RETURN TO CONTINUE
#####')
```

```
                disp('    #####
#####')
                disp('
#########################################################################')
                pause;
            %~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Plot BOTH the Normal and "Exact" Bound Approximations on One Page

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

            figure

        %Plot the Normal approximation results
            clear graph
            axis('square')
            subplot(121)
            plot(phat,L(:,1),'-r',   phat,L(:,2),'-g',   phat,L(:,3),'-b',  ...
                phat,U(:,1),'-r',   phat,U(:,2),'-g',   phat,U(:,3),'-b' );
                axis('square')
                Text_Normal_1 = '95%% Conf. Ints. for P(CC) (Normal Approx.)';
                Text_Normal_2 = '\n Red: n = %3i, Green: n = %3i, Blue: n = %3i';
                Text_Normal = [Text_Normal_1 Text_Normal_2];
                title(sprintf(Text_Normal,n(1),n(2),n(3)))
                legend('red: n = Smallest', 'green: n = Medium', 'blue: n =
Largest','Location','NorthWest')
                xlabel('phat = ML Est. of P(Correct Classif.)')
                ylabel('p = P(Correct Classif.)')
                grid

        % Plot the Exact results
            subplot(122)
            plot(phat,Le(:,1),'-r', phat,Le(:,2),'-g', phat,Le(:,3),'-b',  ...
                    phat,Ue(:,1),'-r', phat,Ue(:,2),'-g', phat,Ue(:,3),'-b' );
                axis('square')
                Text_Exact_1 = '95%% Conf. Ints. for P(CC) (Exact)';
                Text_Exact_2 = '\n Red: n = %3i, Green: n = %3i, Blue: n = %3i';
                Text_Exact = [Text_Exact_1 Text_Exact_2];
                title(sprintf(Text_Exact,n(1),n(2),n(3)))
                legend('red: n = Smallest', 'green: n = Medium', 'blue: n =
Largest','Location','NorthWest')
                xlabel('phat = ML Est. of P(Correct Classif.)')
                ylabel('p = P(Correct Classif.)')
                grid


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
    % Display the Values of Ue and Le for the specific case: P(CC) = 1.0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%

        %-------------------------------------------------------------
        % Display the values of Ue and Le for P(CC) = 1.0, and the various
        % values of n
        %-------------------------------------------------------------

            % Note that phat = 1 for the last element in the phat vector,
            %  which is phat(Npoints)

            UUe=[Ue(Npoints,1) Ue(Npoints,2) Ue(Npoints,3)];
            LLe=[Le(Npoints,1) Le(Npoints,2) Le(Npoints,3)];
```

```
            disp(' ')
            disp('
##########################################################################')
            disp('   #####
#####')
            disp('   #####                 95% CONFIDENCE INTERVALS
#####')
            disp('   #####                 FOR PHAT = 1
#####')
            disp('   #####
#####')
            disp('
##########################################################################')
            disp(' ')
            disp(' ')

            disp('           n         LLe            UUe ')

            format short g              % Set the format of the characters to be
displayed
                                        % to be the best of %g or %f
                                        % formats, depending on the
                                        % variable
            disp(' ')
            disp([ n' LLe' UUe'])       % Display the three quantities


            disp('
##########################################################################')


       %-------------------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%


    end         % End the "if" loop for making plots

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
% end       % End of the M-file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] A. Mucciardi and E. Gose, "A comparison of seven techniques for choosing subsets of pattern recognition properties," *IEEE Transactions on Computers*, Vol. c-20, No. 9, pp. 1023−1031, September 1977.

[2] D. Koller and M. Sahami, "Toward optimal feature selection," *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 284−296, 1996.

[3] Y. Yang and J. Pederson, "A comparative study of feature selection in text categorization," *Proceedings of the Fourteenth International Conference on Machine Learning*, Vol. 14, pp. 412−420, 1997.

[4] B. Bissinger, R. Culver, and N. Bose, "Minimum Hellinger distance based classification of underwater acoustic signals," M.S. Thesis, Pennsylvania State University, University Park, PA, 2009.

[5] H. L. Van Trees, *Detection, Estimation, and Modulation Theory*, Part 1, New York: Wiley, 1968.

[6] G. A. Clark, C. L. Robbins, K. A. Wade and P. R. Souza, *Cable Damage Detection System and Algorithms for the Advanced Development and Process Technologies (ADAPT) Program*, Lawrence Livermore National Laboratory Report LLNL-TR-413970, March 2009.

[7] R. Duda and P. Hart, *Pattern Classification*, Second Edition, New York: Wiley, 2001.

[8] S. Theoridis and K. Koutroumbas, *Pattern Recognition*, Third Edition, Burlington, MA: Elsevier, 2006.

[9] R. V. Hogg and A. Craig, *Introduction to Mathematical Statistics*, New York: Macmillan Publishing Co., 1978.

[10] B. Silverman, "Density estimation for statistics and data analysis," *Monographs on Statistics and Applied Probability*, London: Chapman and Hall, 1986.

[11] M. Girolami and C. He, "Probability density estimation from optimally condensed data samples," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 9, pp. 1253−1264, 2003.

[12] H. Chen and P. Meer, "Robust computer vision through kernel density estimation," *ECCV*, Copenhagen, Denmark, Vol. 1, pp. 236−250, 2002.

[13] E. Parzen, "On Estimation of a Probability Density Function and Mode," *Annals of Mathematical Statistics*, Vol. 33, pp. 1065−107, 1962.

[14] D. Specht, "Probabilistic Neural Networks," *Neural Networks*, Vol. 3, pp. 109−118, 1990.

[15] J.B. Cain, "An improved probabilistic neural network and its performance relative to other models," *SPIE*, Vol. 1295, pp. 354−365, 1990.

[16] P. Narendra and K. Fukunaga, “A branch and bound algorithm for feature subset selection,” *IEEE Transactions on Computers*, Vol. c-26, No. 9, pp. 917−922, September 1977.

[17] C. Lee and D. Hong, “Feature Extracting using the Bhattacharyya Distance,” *Systems, Man, and Cybernetics*, Vol. 5, pp. 2147−2150, 1997.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. R. Clark Robertson
   Naval Postgraduate School
   Monterey, California

4. Grace A. Clark
   Naval Postgraduate School
   Monterey, California

5. Monique P. Fargues
   Naval Postgraduate School
   Monterey, California

6. Matthew J. Wilder
   Naval Postgraduate School
   Monterey, California